Escuela Politécnica Superior

El malware y las redes sociales

Máster CiberSeg

Máster Universitario en Ciberseguridad

Trabajo Fin de Máster

Autora:
Melanie Mariam Cruz Morgado

Tutor:
Rafael Ignacio Álvarez Sánchez

Septiembre 2021

Universitat d'Alacant
Universidad de Alicante

Agradecimientos

Este pequeño espacio se lo dedico a todas esas personas que me han acompañado y directa o indirectamente han puesto su granito de arena para que me fuese posible afrontar la realización de este trabajo y superar el resto de este año tan distinto en muchos aspectos.

No puedo empezar de otra manera que haciendo un agradecimiento especial a mi tutor, Rafa, por toda su comprensión, paciencia y dedicación. De verdad que ha sido muy agradable trabajar conjuntamente, tanto que he querido repetir tutor del trabajo de fin de grado y me siento muy agradecida de que también aceptara tutorizarme el trabajo de fin de máster, a pesar de tenerme que aguantar dos años seguidos.

En estas líneas no pueden faltar unas palabras dedicadas a mi familia, a mis compañeros del máster, a mis amistades… En definitiva, todo aquel que me ha apoyado incondicionalmente en todo momento, algo que ha sido esencial para conseguir ese ánimo y motivación necesarios, que no siempre se consiguen por una misma.

Por último, pero no menos importante, agradecer también la comprensión y el esfuerzo de cada profesor del máster en toda su labor, a pesar de las circunstancias en las que nos ha tocado cursarlo y prácticamente solo conocernos a través de las pantallas.

De manera similar ocurrirá con la defensa de este trabajo, ya que a los integrantes del tribunal los veré en una videollamada y, aunque escribiendo esto no sepa a quienes les ha tocado formar parte de él, estoy segura de que serán justos en la corrección y quiero agradecerles desde aquí todo el tiempo dedicado en ella.

Personalmente, este trabajo no solo significa dar fin al máster en ciberseguridad, sino también dar por primera vez un paso fuera del camino que he seguido año tras año de recorrido académico. Para mí es algo muy especial y me alegra haberlo podido compartir con todas estas personas, aunque últimamente haya tocado que fuese de manera telemática en la mayoría de los casos, pero nunca he dudado que estaban ahí.

Por todo ello, muchísimas gracias.

“La desconfianza es la madre de la seguridad”

Aristófanes

“Ya no buscamos las noticias, ellas nos encuentran”

Erik Qualman

“Solo después de que los usuarios hayan sido engañados,

realmente prestarán atención a la capacitación”

Todd Fitzgerald

Índice general

Índice de tablas

1. Introducción

En la actualidad, el *malware* se ha convertido, en cierta forma, en una industria que puede generar beneficios o impactar negativamente en la actividad informática de muchas empresas y servicios. Además, sigue evolucionando y las redes sociales también están comenzando a jugar un papel importante en todo esto, por la colaboración en su propagación y éxito en algunos casos.

Asimismo, es evidente que las redes sociales, siendo también relativamente nuevas, en poco tiempo han conseguido ocupar un puesto en la sociedad a todos los niveles, tanto personalmente como profesionalmente, influyendo en el día a día de las personas e incluso en sus decisiones u opiniones, a causa de la creciente cantidad de tiempo que pasan en ellas.

En relación con la ciberseguridad, sustenta el hecho de que en un sistema el elemento más vulnerable suelen ser las personas. Aunque, no se pretende buscar solo la seguridad y privacidad del sistema, siendo en este caso la red social, sino de los usuarios de la misma y evitar que estos sean engañados o manipulados.

1.1 Propuesta

El proyecto que propongo consiste en analizar distintos tipos de *malware* y las implicaciones de su uso en conexión con las redes sociales existentes, permitiéndome probar y diseñar *software* relacionado con ello como fruto de dicho análisis.

Más concretamente, se realizará el análisis y pruebas necesarias para diseñar una API que pueda ser usada como herramienta para identificar la presencia de *botnets* maliciosas u otros tipos de fraudes en las redes sociales que se vayan incluyendo en esta. Comenzando, en este caso, por dos muy conocidas y usadas por la sociedad: Twitter e Instagram.

1.2 Motivación

El origen de este proyecto parte del gran interés que me despiertan las redes sociales, habiendo enfocado también mi trabajo de fin de grado en ellas, aunque de distinta forma.

Las redes sociales son un mundo muy útil, llamativo y peligroso a partes iguales. Es por ello que tienen una relación clara con la seguridad, en gran medida porque forma parte de nuestro día a día y a estas puede acceder prácticamente cualquier público. Es decir, son muy accesibles, por lo que un fallo en su seguridad podría afectar en cuestión de segundos a una gran cantidad de personas.

Pero la seguridad informática es un campo muy amplio, así que he querido centrarme en el *malware*. Durante el máster, hay poco temario enfocado especialmente a este tema y a mí siempre me ha llamado mucho la atención, sobre todo por el hecho de que es algo a lo que está expuesta cualquier persona que tenga acceso a un dispositivo.

Por todo ello, resumiría este trabajo en un estudio de la relación entre la ciberseguridad (*malware*) y el día a día de las personas (redes sociales).

1.3 Objetivo

El objetivo principal de este trabajo es realizar una investigación sobre la relación entre el *malware* y las redes sociales. Lo cual nos lleva a las *botnets* sociales, las cuales explicaré y expondré ejemplos significativos, cómo se descubren y cómo se evitan; siempre y cuando sea posible acceder a dicha información ya que, en muchas ocasiones, al ser un tema sensible, no se encuentra de manera pública.

Además, después de estudiar de mejor manera el *malware* y, en concreto, las *botnets* sociales e indagar y hacer pruebas en alguna red social conocida, será posible comprender cómo de sencillo puede llegar a ser encontrar, difundir y, lo más importante, identificar la presencia de las mismas, ya que vulneran la seguridad y privacidad de los usuarios. Por lo que se pretende crear alguna herramienta que ayude a detectarlo, aportando datos con los que sea posible identificar una *botnet* social, siendo este el primer paso antes de actuar sobre ellas.

1.4 Justificación

Con esta idea, pretendo realzar la importancia de que las redes sociales mejoren día a día su seguridad informática y que deberían dedicar gran parte de sus esfuerzos a ello, ya que son sitios muy frecuentados por la población hoy en día. Esto no quiere decir que actualmente no se esté trabajando para que así sea, pero siempre se debe avanzar y mejorar, y yo quería poner mi granito de arena en ello.

Además, mientras más influencia tenga un sistema, más peligroso puede ser, siendo este el caso de las redes sociales. De hecho, los estafadores confían cada vez más en la popularidad y riqueza de estas para cometer actos fraudulentos y, a pesar de ello, en prácticamente ninguna red social se presta la atención necesaria en examinar el contenido en busca de *spam*.

2. Estado del arte

Malware es la abreviación de *malicious software*, es decir, *software* malicioso. Esto significa que es un programa que se ejecuta en un dispositivo, pero en lugar de servir para un propósito útil, su función es causar daño en el mismo, pudiéndose manifestar de diversas maneras. Además, es relativamente nuevo, ya que hace menos de medio siglo que se ideó y en un principio fue una forma de vandalismo o travesura, pero poco a poco ha ido evolucionando a una motivación superior, principalmente la obtención de un beneficio económico.

En este apartado se analizarán algunos *malware*, haciendo especial hincapié en aquellos que hayan sido más conocidos, relevantes, dañinos, nuevos y/o los que hayan utilizado las redes sociales de alguna manera.

2.1 Tipos de malware

Los equipos están expuestos a muchas amenazas diferentes, así que a continuación se exponen los distintos tipos de *malware* existentes, para empezar a familiarizarnos con el concepto, además de algunos ejemplos de la mayoría de ellos. [1]

2.1.1 Virus

Un virus es un programa capaz de autorreplicarse. Una vez que haya logrado infectar un equipo, este puede propagarse entre los archivos del mismo o de su misma red. Además, se propagan de manera muy sencilla por Internet mediante la descarga de archivos, algo que prácticamente todos hacemos con mucha frecuencia.

Su objetivo es infectar partes de un dispositivo para alterar el correcto funcionamiento del mismo, pero necesita que un usuario lo ejecute. Este suele pensar que se trata de una aplicación legítima, lo acciona e inocentemente le permite replicarse e infectar el equipo. Aunque también hay otras herramientas que consiguen camuflar los virus dentro de otros tipos de archivos para ejecutarlos de forma indirecta. Con ello no persiguen una finalidad económica directa, solo se encargan de dañar el sistema del equipo y/o hacerlo más lento.

En muchas ocasiones se utiliza indistintamente la palabra “virus” y la palabra “*malware*”, pero esto es técnicamente erróneo, porque los “virus” no son lo mismo que los “*malware*”, sino un tipo de estos, por lo que un “virus” es un “*malware*”, pero un “*malware*” no necesariamente es un “virus”. Por ello, se debe tener cuidado a la hora de buscar información al respecto, ya que puede ser equivocada y confundirnos en los conceptos.

2.1.1.1 Brain

Este virus de ordenador fue ideado en 1984 y finalizado en septiembre de 1986, hace ya 25 años, bajo el nombre de “Brain”. El nombre se debe a que su intención era evitar que se usase una copia ilegal del programa que creaban y comercializaban dos hermanos de Pakistán en la tienda Brain Computer Services. Este infectaba el sector de arranque de los discos Floppy y no fue detectado de manera pública hasta 2 años después de su lanzamiento. [2]

En su origen no era común que causara daños, cosa que cambia en versiones posteriores del virus, las cuales sí pueden llegar a borrar una cantidad importante de datos del disco duro. De todos modos, la versión original causó especiales problemas en los dispositivos con MS-DOS como sistema operativo. Ocurría porque al alojarse en el *boot* o sector de arranque de los disquetes, cuando estos eran inferiores a la versión 2.0, al tener menor capacidad, el virus infectaba el disco duro de la computadora y también el propio disquete, provocando el borrado de pequeñas cantidades de datos cuando este estuviese prácticamente lleno.

Una peculiaridad notable de este virus fue su carta de presentación, ya que antes de realizar la infección mostraba lo siguiente: "*Welcome to the Dungeon ... (c) 1986 Brain & Amjads (pvt) Ltd ... 430791.443248.280530 VIRUS_SHOE RECORD V9.0 ... Dedicated to the dynamic memories of millions of virus who are no longer with us today - Thanks GOODNESS !! ... BEWARE OF THE er ... VIRUS...*". [3]

Con este mensaje lo que pretendía era hacer ver al usuario que estaba infectado con un virus, pero sin intenciones destructivas; sino que, al usar una copia pirata de su programa, se pudieran poner en contacto con ellos para eliminar el virus, ya que son los datos reales de los creadores del mismo. De hecho, han recibido muchas llamadas, siendo la primera la de la Universidad de Miami, lo cual les sorprendió, ya que su método de propagación era más costoso y lento que hoy en día, al no propagarse por Internet, sino por disquetes que transportaban las personas por todo el mundo, muchas veces sin ser conscientes de ello.

También es relevante que, gracias a su buena intención, a pesar de haber sido los creadores del primer virus informático y hacerlo sin esconderse, aseguran que no han recibido ninguna amenaza ni desafío. Además, aunque les duele ver que de algo que crearon sin intención destructiva se haya convertido actualmente en actos delictivos, no se arrepienten de haberlo creado, ya que piensan que era algo inevitable que hubiese ocurrido independientemente de que ellos lo crearan o no.

De todos modos, este virus ya ha sido erradicado por completo. La Corporación IBM creó BrainStop, porque afectaba a sus plataformas, y lo pusieron en el mercado llegando a alcanzar el top ventas de *software*.

Aunque después se supo que había otra forma más económica de eliminar el virus, ya que bastaba con reiniciar el sistema con un disquete de arranque que no contuviese el virus en la unidad A y el infectado en la B. De esta manera, al ejecutar el comando "A:\SIS B: [ENTER]", se borraba el virus del disco infectado.

2.1.1.2 Antipiratería

Siguiendo en la línea de la antipiratería, este 2021 la empresa Sophos ha lanzado el llamado "*malware* antipiratería", sin tener nombre propio, al menos por el momento. [4]

Este virus se hace pasar por un videojuego y el hecho de que sea antipiratería es porque su único objetivo es impedir que el usuario infectado ingrese en páginas de descarga de contenidos y *software* de manera ilegal, es decir, sin contar con una licencia. Por lo tanto, no pretende robar información ni recursos del usuario, ni suplantarlo, sino que es como una denegación de servicio al bloquear dicho acceso ilegal.

Además, este ha tenido una relación directa con las redes sociales, ya que se ha distribuido a través de Discord, lugar por donde se enviaba como ejecutables y después se escondía en el equipo como copias pirateadas de paquetes *software*. De todos modos, esta no fue la única forma de distribución, ya que también se ha hecho uso de BitTorrent, un servicio de intercambio de archivos, en el que se podía encontrar este virus bajo la imagen de *software* pirata y el nombre de un videojuego famoso, herramienta de productividad y/o productos de seguridad.

En ambos casos, son archivos que han sido comprimidos con uno de texto y otros, entre los que se encuentra el acceso directo que indica ThePirateBay. Al hacer doble *click* aparece una ventana con un mensaje de "Error del sistema" y expone que no ha sido encontrado un archivo .dll en el ordenador, así que arregla el problema reinstalando el programa.

En el caso de tenerlo, la manera de eliminarlo sería limpiar el archivo *C:\Windows\System32\Drivers\etc\hosts* de forma manual, ejecutando como administrador para tener los permisos de eliminar todas aquellas líneas que comiencen con "127.0.0.1" y las que mencionen las webs ThePirateBay u otras de piratería.

2.1.1.3 BitCoinMiner

El nombre de BitCoinMiner está relacionado con varios virus que realizan criptosecuestros, es decir, interceptan equipos de usuarios para minar criptomonedas de manera secreta e ilícita. Según el nombre, puede pensarse que la moneda que minan son Bitcoins, siendo esta la criptomoneda más conocida, pero las que en realidad minan son algunas menores como Monero e incluso Ethereum, con las que sí es viable aprovechar las especificaciones del *hardware* ordinario, ya que es el motivo esencial por el que se han creado estos virus en los últimos años.

La minería de criptomonedas, es decir, la criptominería, es la resolución de problemas matemáticos usando equipos informáticos, bajo una gran inversión de dinero para obtener beneficios realizando estos procesos, ya que se necesitan equipos potentes y se gasta una cantidad excesiva de electricidad. A causa de ello y que el uso de las criptomonedas está en auge, ha habido ciberdelincuentes que han creado estos virus para usar equipos de otros usuarios e inyectarles herramientas maliciosas en el sistema para poder hacer uso de procesadores (CPU) y tarjetas gráficas (GPU) para realizar dicha minería, pudiendo consumir hasta la totalidad de los recursos del sistema. Es por eso que el usuario puede identificarlo, porque el equipo prácticamente deja de ser útil, llegando a bloquearse y/o sobrecalentarse en exceso por sobrecarga, es decir, atacan a la disponibilidad del equipo, siendo uno de los pilares básicos de la seguridad. Además, como consecuencia colateral a esto, es posible que incluso se pierda de forma permanente la información que se tenga en el equipo.

Su forma de distribuirse es una de las vías que usan la mayoría de virus, mediante *spam*, es decir, correos basura con archivos maliciosos, aunque a primera vista no lo parezcan y por eso el usuario suele abrirlo y es cuando se descarga e instala el virus en el equipo. [5]

2.1.1.4 I love you

"I love you" es conocido como el virus del amor, es uno de los más conocidos y de los que más daño causó en su momento. Su asignación en esta categoría no es exacta, porque se trata de un virus de macro con una línea difusa con el gusano, un tipo que será explicado a continuación a este; esto ocurre a causa de su capacidad para reproducirse mediante las redes electrónicas modificando los ficheros en el dispositivo infectado. De todos modos, me ha parecido más correcto incluirlo aquí por el hecho de que necesite la ejecución humana para activarse y propagarse.

Su funcionamiento comienza cuando el usuario abre un correo electrónico cuyo asunto es "I Love You" y contiene un archivo adjunto llamado "LOVE-LETTER-FOR-YOU.TXT.vbs", simulando una carta de amor, cuando en realidad contiene el código malicioso. Este sobrescribe los archivos con extensiones .vbs y .vbe, elimina los archivos .js, .jse, .css, .wsh, .sct y .hta y sustituye los archivos .jpg, .jpeg, .mp3 y .mp2; crea otros con el mismo nombre, pero cuya extensión sea .vbs e introduce el código malicioso. De esta manera, cuando el usuario abra cualquiera de esos archivos infectados, se mandará un correo electrónico a sus contactos con las mismas características y así sucesivamente.

Este *malware* atacó a varias empresas conocidas, quienes tuvieron que tomar medidas al respecto. Por ejemplo, empresas de tecnología como Vodafone, Siemens o DELL, medios de comunicación y periódicos como El País, la Cadena Ser, La Razón y ABC, e incluso la línea aérea Iberia. [6]

2.1.2 Gusano

El gusano es un tipo de *malware* ligeramente diferente al virus, pero al igual que este, puede autorreplicarse, con la peculiaridad de que es capaz de enviarse a sí mismo muchas veces de manera automática, es decir, sin necesidad de la interacción de un usuario. De esta manera, consiguen afectar a un gran número de dispositivos rápidamente, llegando a ser una gran amenaza.

Más específicamente, de los gusanos informáticos que se conocen hasta el momento, la mayoría se propagan mediante uno o varios métodos de los siguientes: archivos adjuntos a correos electrónicos o mensajería instantánea; enviando enlaces que accedan a un recurso web o FTP, enlaces en un mensaje ICQ o IRC o compartiendo archivos P2P en redes de uso; usando partes de un sistema operativo que sean automáticas e invisibles para el usuario; disfrazándose de paquetes de red para introducirse en la memoria del equipo directamente; o explotando errores en la configuración de la red u otros agujeros de seguridad que puedan encontrarse tanto en el sistema operativo como en las aplicaciones, con el fin de autocopiarse en un disco que sea totalmente accesible.

Su función principal es infectar otros equipos mientras permanece activo en todos los sistemas infectados. Aunque no tienen por qué ser maliciosos, de hecho, hay ocasiones en los que simplemente se utilizan para averiguar si es posible violar la seguridad del sistema; pero en caso de que sí sean maliciosos, el código que contienen para llevar a cabo el ataque se denomina carga útil.

Este tipo de *malware* tiene un papel importante en la creación de *botnets*, ya que se suelen utilizar para ello. Estas serán explicadas más adelante, pero se adelanta que son redes de dispositivos *zombies* que tienen la capacidad de actuar de manera simultánea cuando un operador les da la orden. Varios ejemplos serían el envío masivo de *spam*, *malware* u otros tipos diferentes de ataque informático, como el de denegación de servicio, ya que pueden saturar/colapsar la red al crear un sinfín de comunicaciones entre los dispositivos.

2.1.2.1 Morris

El gusano Morris, habiendo sido lanzado en 1988, se convirtió en el primer gusano informático conocido. El propósito de Robert Morris, un estudiante estadounidense, era descubrir cómo de grande era realmente Internet, pero su código contenía muchos errores, lo cual causó una variedad de problemas en los *hosts* afectados. Finalmente, el resultado fue que miles de ordenadores UNIX se sobrecargaron y el daño financiero osciló entre los 10 y 100 millones de dólares. [7]

2.1.2.2 SQL Slammer

El gusano Slammer no usó ningún tipo de propagación común de las explicadas, lo cual lo hizo único. Este generaba una serie de direcciones IP aleatorias a las que envió el *malware*, esperando que no se defendiesen y también tuvo una veloz propagación a través de sistemas que utilizaban una versión específica de SQL, afectando incluso al servicio de emergencias de EEUU. [8]

Su finalidad fue que todo equipo infectado no lo supiera, pero que participara en ataques DDoS de sitios webs relevantes, usando una vulnerabilidad de desbordamiento de buffer en distintos productos de Microsoft SQL Server que no se hubiesen instalado el Service Pack 3. El procedimiento era que cada equipo infectado mandase un paquete de unos 380 bytes al puerto 1434 UDP de manera inmediata, consiguiendo a unos 10 minutos de su lanzamiento, en enero de 2003, infectar a miles de servidores y que colapsara Internet de manera global.

2.1.2.3 Storm

Storm Worm se trata de un correo electrónico que recibían las víctimas con un informe falso de noticias sobre una ola de tormentas, de ahí el nombre, apelando a que ya había matado a cientos de personas en toda Europa.

Según los datos de la fuente utilizada, desde 2007 que fue su lanzamiento hasta los 10 años posteriores, logró mandar más de 1200 millones de estos correos electrónicos, aunque se cree que todavía hay ordenadores infectados sin que sus propietarios lo sepan, formando parte de esta *botnet*. [7]

2.1.2.4 Stuxnet

Stuxnet fue el primer gusano informático creado por las agencias de inteligencia de Israel y EEUU con un objetivo militar específico: vulnerar la central nuclear iraní de Natanz con el fin de sacudir las centrifugadoras para enriquecimiento de uranio y descarrilar o, al menos, retrasar el programa nuclear de Irán.

El experto israelí Shimon Gruper asegura que Stuxnet cumplió su misión de manera estricta y que nunca fue diseñado para destruir dicho programa nuclear. Él dice que “su objetivo era ir causando poco daño en un periodo largo de tiempo. Siendo esa una de las características que lo convierten en uno de los más inteligentes, junto a su discreción y la dificultad de identificarlo ya que, si causara un gran daño e inmediato, habría sido rápidamente neutralizado”. [9]

Este gusano fue identificado por primera vez por la comunidad Infosec en 2010, pero su desarrollo probablemente comenzó en 2005, aproximadamente.

A pesar de su incomparable capacidad de propagación y su tasa de infección generalizada, Stuxnet hace poco o ningún daño al dispositivo que no participe en el enriquecimiento de uranio. Cuando infecta una computadora, verifica si está conectada a modelos concretos de controladores lógicos programables (PLC) fabricados por la empresa Siemens. Estas computadoras consiguen interactuar gracias a estos PLC y es así como controlan la maquinaria industrial, entre la que se encuentran las centrifugadoras de uranio.

Lo que consigue el gusano es alterar la programación de los PLC, haciendo que estas centrifugadoras giren tan rápido y durante tanto tiempo, que acaben dañando o destruyendo el delicado equipo en el proceso. Mientras esto sucede, los PLC le dicen a la computadora del controlador que todo está funcionando bien, lo que dificulta la detección o el diagnóstico de lo que va mal hasta que es demasiado tarde. [10]

2.1.3 Troyano

El troyano es un tipo de *malware* que se caracteriza por ir escondido dentro de un programa legítimo o disfrazado de él o de otro archivo, con la intención de introducirse en el equipo de forma secreta, simulando el uso de un Caballo de Troya, siendo este el motivo por el que ha sido nombrado así.

Este debe lograr pasar desapercibido mientras accede al equipo víctima, con el fin de ejecutar acciones ocultas que abran una puerta trasera para que atacantes u otros programas maliciosos puedan acceder y/o robar información, comúnmente bancaria; pero en sí mismo no causan daños en primera instancia y tampoco suelen autorreplicarse.

2.1.3.1 Zeus

Zeus también es uno de los *malware* más conocidos en este “mundillo” y, desde que se publicó su código fuente en 2011, la mayoría de troyanos bancarios tienen influencia de este. Incluso cientos de personas han sido encarceladas por participar en estafas en las que se hacía uso de este troyano.

Aunque apareció en 2007, después de un ataque contra el Departamento de Transporte de EEUU, siendo solo el principio; ya que desde entonces infectó 10 millones de equipos y robado cientos de millones de dólares, entre los que se incluyen bancos, agencias gubernamentales e instituciones públicas, hasta que sus creadores revelaron el código fuente años después.

Su popularidad fue tal, que también se extiende a una versión móvil llamada ZitMo, que es capaz de evadir los sistemas de doble verificación que envían el código de acceso a través de un SMS. [11]

2.1.3.2 Tetrade

Tetrade es una familia de 4 troyanos bancarios creados, desarrollados y propagados por ciberdelincuentes de Brasil que operan en América Latina y en Europa Occidental, siendo España el principal país europeo objetivo de este. Además, hoy en día sigue siendo una amenaza a pesar de que ya se han llevado a cabo varias detecciones de personas relacionadas.

Su operativa es como un proyecto de MaaS, es decir, *malware* como servicio, usando *scripts* Autolt o VBS añadidos en archivos MSI que ejecutan DLL maliciosas mediante la técnica DLL-Hijack; y su objetivo es eludir las soluciones de seguridad para robar las contraseñas y proporcionar acceso remoto para capturar la conexión por Internet a los bancos o robar carteras de criptomonedas. [12]

2.1.3.3 Apps para Android

El sistema operativo Android tampoco se libra de los troyanos. Este mismo 2021, Google Play ha debido eliminar 9 aplicaciones que escondían troyanos y ya habían llegado a los 6 millones de descargas aproximadamente. Aunque eso no significa que se hayan erradicado, ya que pueden seguir disponibles en plataformas de terceros y/o de descarga.

Sus nombres son: Processing Photo, App Lock Keep, Rubbish Cleaner, Horoscope Daily, Horoscope Pi, App Lock Manager, Lockit Master, Inwell Fitness y PIP Photo. Todas ellas contaban con una variante de Android.PWS.Facebook.15, un troyano que usa formatos de archivo y *scripts* de Java para robar las credenciales de acceso y las contraseñas de la red social Facebook. Aunque podrían haberse obtenido credenciales de cualquier otro sitio web. [13]

2.1.4 Ransomware

Este tipo de *malware* seguramente sea del que más se ha hablado estos últimos años por su salto a las redes corporativas conocidas de grandes empresas y organizaciones, siendo estas las que suelen ser más cuidadosas, pero a la vez las que más influencia financiera tienen y, por lo tanto, de las que más dinero se puede explotar. De todos modos, no es este el motivo de su popularidad exactamente, sino que dichos casos suelen ser los más mediáticos; pero atacan a pequeños equipos, ya que los *malware* de este tipo no modulan y en general actúan prácticamente de cualquier dispositivo a otro, pudiendo llegar a uno o varios que formen parte de una empresa y, por supuesto, en ese caso se pueden aprovechar más de estos.

La evolución del Ransomware fue tal que, en 2017, pasó de registrarse de media un ataque cada 2 minutos a hacerlo cada 40 segundos, es decir, se triplicaron los casos. Su objetivo es secuestrar datos del mayor número posible de equipos para bloquear completamente el acceso a ellos, a no ser que se pague un rescate para desbloquearlos. [14]

Hay algunos simples que solo bloquean el sistema siendo sencillos de desbloquear; otros en cambio usan la técnica de extorsión viral criptográfica que encripta los archivos de las víctimas haciéndolos virtualmente inaccesibles. También existe el riesgo de perder todos esos datos, ya que el método de actuación suele ser aumentar la cantidad monetaria a pagar a medida que va pasando el tiempo y si no se paga antes de la fecha acordada, los atacantes eliminarán todos esos datos. De todos modos, en esos casos no hay garantía de que vayan a cumplir con su palabra al realizar el pago y hay que tener especial cuidado, ya que no solo está en riesgo la economía de la organización atacada, sino también la información de millones de personas.

Además, aunque no sea el objetivo principal ni lo más común, también es posible que los ciberatacantes hagan pública la información, ya que estos cuentan con la clave de cifrado y sí pueden acceder a ella, como un añadido a la amenaza. No obstante, esto no suele interesar en la mayoría de casos, ya que la información no suele ser de interés público o excesivamente comprometedora, mientras no sea de alguna gran empresa, por ejemplo.

Respecto a la manera de propagación, suele ser mediante otros tipos como troyano o gusano, haciéndolos especialmente dañinos y, como ya se ha comentado, cada vez más comunes y frecuentes.

2.1.4.1 Maze / ChaCha

Este ransomware es conocido tanto por el nombre Maze como ChaCha y está en la primera posición de entre los de este tipo de *malware*, ya que ha llegado a ser el responsable de un tercio de los ataques informáticos y también fue el creador de la técnica de robar los datos antes de cifrarlos, para que en caso de que la víctima se negara a pagar el rescate, poder amenazarla con publicarlos.

REvil y DoppelPaymer son dos de los que siguieron la manera de extorsionar de este ransomware como ejemplo. Siendo el primero, REvil, también conocido como Sodin o Sodinokibi, el que actualmente tiene el récord de la demanda de rescate más alta vista hasta ahora: 50 millones de dólares a la empresa tecnológica Acer, realizado en marzo de 2021.

2.1.4.2 Conti

Conti, también conocido como el ransomware IOCP, lleva activo desde finales de 2019 y tiene un detalle que lo hace especial. Los atacantes, además de desbloquear los datos cuando la víctima realice el pago, también le ofrecen ayuda para cerrar el agujero de seguridad por el que consiguieron entrar y les ayuda a mejorar más su seguridad recomendando un *software* que dificulta el trabajo a los ciberdelincuentes.

2.1.4.3 Netwalker / Mailto

Netwalker o Mailto, indistintamente, es un ransomware que ya no sigue activo desde la operación policial de enero de este 2021, en la que la policía se apoderó de los recursos de este ransomware en la *dark web*, es decir, la web oscura donde los sitios no están indexados.

Sus creadores ofrecieron el alquiler de este ransomware a estafadores que, de acuerdo con *Bleeping Computer*, el distribuidor podía conseguir el 70% del rescate. Además, como fue algo tan llamativo, los ciberdelincuentes lo demostraron con capturas de pantalla de grandes transferencias bancarias. Asimismo, en algunos casos creaban un sitio web para publicar de manera automática los datos robados, después del límite establecido por el atacante, si no ha recibido su recompensa.

2.1.5 Scareware

El scareware, según su nombre indica, es un *software* de miedo o susto, es decir, programas que muestran una alerta que aparenta ser bastante grave, pudiendo ser por ejemplo un problema o una actualización.

El propósito es generar miedo y hacerte creer que a tu equipo le ocurre algo grave, así que invitan a comprar la solución o dan un número de teléfono para llamar, siendo todo esto una estafa. En la mayoría de los casos, no le ocurrirá nada al dispositivo, pero la finalidad es que pagues por dicha solución, que en realidad no es necesaria. En esta categoría entran los antivirus falsos.

Otro caso de scareware es asustar mediante correo electrónico, con el objetivo de obtener información personal de la víctima. En el correo indican que hay algún problema que se debe resolver lo antes posible y para ello requieren que introduzca unos datos personales, pero lo que realmente hacen con ellos es suplantar la identidad del usuario, incluirlo en campañas de *spam* y/o vender esos datos a terceros; vulnerando así la confidencialidad de los datos personales. [15]

2.1.6 Spyware

El tipo de *malware* spyware se asemeja al troyano, aunque en este caso el ciberatacante “solamente” espía el dispositivo de manera silenciosa, en vez de permitir el acceso a él, pero puede robar información bancaria como los números de tarjeta de crédito y cualquier otra información personal y confidencial que esté almacenada o sea accesible desde el equipo.

La característica principal es que no intenta molestar, sino que está diseñado para pasar desapercibido para no ser detectado, ya que busca espiar para sacar la mayor información posible, extrayendo datos a los que no está autorizado, como los comentados anteriormente, o también con algún otro fin siniestro, como espiar por la cámara web.

Además, puede ser instalado en un dispositivo mediante la interacción de otra aplicación que lo lance sin darse cuenta o también por sí solo.

2.1.6.1 Zlob

Zlob es un spyware que usa las vulnerabilidades de los códec de ActiveX para infiltrarse en el equipo y comenzar a guardar un registro tanto de las búsquedas y navegación como de las pulsaciones realizadas mediante el teclado. Esta acción es lo que comúnmente se denomina "keyloggers". [16]

2.1.6.2 TIBS Dialer

En el caso del spyware TIBS Dialer, lo que secuestraba era módems, desconectando el ordenador víctima de la línea telefónica de la red local y, en su lugar, lo conectaba a un número de pago diseñado para acceder a páginas pornográficas. [16]

2.1.7 Adware

El adware es un tipo de *malware* muy similar al spyware, aunque no siempre es tan malicioso. En este caso, al instalarse en el equipo, intenta mostrar publicidad o anuncios de relevancia y de una manera agresiva. Muchos de estos muestran tantos anuncios que pueden ralentizar el equipo infectado y, por lo tanto, cualquier búsqueda o actividad que se realice en él.

De todos modos, no siempre es dañino para el equipo y es por ello que está al borde del hecho de considerarse como *malware*. Su único objetivo es entrar en un dispositivo y comenzar a mostrar todo ese *spam*, ya sea en forma de *popup*, es decir, una ventana emergente en momentos aleatorios, mientras se navega por Internet o durante la ejecución de un programa. Además, la manera en la que normalmente accede al equipo es a través del proceso de instalación de otras aplicaciones, casi como si fuera un troyano pero sin dañarlo.

Llena el equipo de avisos publicitarios que en la mayoría de los casos la publicidad puede ser engañosa. Mayormente afectan a los navegadores cambiando, por ejemplo, la página de inicio y luego bloquean los intentos de cambiarla. También abren ventanas aleatoriamente y redirigen a otros sitios. Todo ello con la finalidad de obligar a usar sus servicios, visitar sus sitios web, ver sus anuncios o incluso realizar una estafa, para ganar dinero extra a costa de la víctima.

2.1.8 Bloatware

Bloatware no necesariamente es un *malware*, pero puede llegar a serlo, ya que es todas aquellas aplicaciones no esenciales preinstaladas por el fabricante en un equipo, no siendo útiles siempre y creando un problema, en muchas ocasiones, cuando no pueden ser desinstaladas de forma sencilla.

Eso ocurre porque, por ejemplo, en el caso de los dispositivos móviles habría que rootearlos, siendo un proceso complejo y sensible, por lo que hay que tener conocimientos para ello. Esto se traduce en recursos y espacios desaprovechados, por lo tanto, ataca a la disponibilidad del equipo, siendo este uno de los pilares de la seguridad de la información.

Es tan sencillo comprobarlo que solo ha sido necesario revisar un *smartphone* y ver qué aplicaciones no se podían borrar. En este caso, un Xiaomi con el "Navegador Mi" que, por ejemplo, sería una aplicación bloatware, ya que no permite ser eliminada, a pesar de tener el buscador de Google y no necesitar dos aplicaciones con la misma función.

Asimismo, aunque no es lo habitual, porque los fabricantes intentan que sean seguras para los equipos, en algunas de estas aplicaciones se ha llegado a detectar la presencia de *malware*.

2.1.9 Rootkit

El rootkit es un tipo especial de *malware*, ya que ha sido creado para ocultar la presencia de otro *malware* en el sistema del equipo infectado, además de aportarle a los ciberdelincuentes permisos de administrador en el mismo.

2.1.10 Riskware

Riskware es todo *software* legítimo que puede provocar daños en el equipo si es utilizado con fines ilícitos por los ciberdelincuentes con el objetivo de eliminar, bloquear, modificar o copiar los datos, y afectar al rendimiento del equipo o su red. Además, es un peligro que la mayoría de los anti-*malware* no suelen detectar el *software* de riesgo, es decir, qué aplicaciones representan una amenaza para el equipo, por lo que se suele detectar por la disminución significativa del rendimiento.

En realidad, este tipo se ha incluido aquí, pero hay controversias de si es o no un *malware*, ya que hay aspectos en los que se puede considerar como tal, pero el *software* no está diseñado para serlo, sino que sus debilidades son explotadas.

2.1.10.1 Control remoto

Un ejemplo podrían ser los programas de control remoto. [17] Estos, en ocasiones son utilizados por administradores de sistemas y servicios de asistencia técnica, incluso dentro de empresas corporativas, para diagnosticar el problema a distancia e incluso poder solucionarlo.

El problema radica en que, si un ciberdelicuente instala en un equipo un programa de ese tipo sin el consentimiento del usuario, dispone de acceso remoto teniendo todo el control. De hecho, Kaspersky Lab ha recopilado casos en los que ha ocurrido esto haciendo uso de WinVNC.

2.1.10.2 SergHelper

El riskware SergHelper [18] afectó a los dispositivos móviles de Apple y su objetivo era evadir el proceso de revisión de la App Store de Apple. Fue eliminada por la empresa, ya que podía instalar versiones de aplicaciones iOS que fuesen modificadas, cuya seguridad había sido garantizada previamente, aprovechándose de los certificados usados para firmar y distribuir aplicaciones cuyo código no fuese revisado, lo que puede suponer un gran riesgo.

2.1.10.3 WhatsApp Plus

Este fue el riskware más notable de Android, cuyo propósito era robar todos los datos que le fuese posible de un teléfono móvil con dicho sistema operativo. Esto lo conseguía proporcionando un enlace de descarga a una actualización a WhatsApp Plus que requería, pero redirigía a una página web que se encontraba escrita completamente en árabe. [18]

2.1.10.4 Clientes de BitTorrent

Como ya se ha comentado en otros tipos, los clientes de BitTorrent son comúnmente usados para difundir *malware*, aprovechándose de que el usuario quiere descargarse de manera "pirata", es decir, sin permisos, algún contenido protegido por derechos de autor, por lo que ya son archivos ilegales y son fácilmente convertibles en *malware*. [18]

2.2 Botnets

Las *botnets* como tal no son un *malware*, sino un grupo de equipos destinados a realizar una o varias tareas de forma repetida y automatizada (*bots*) con una determinada finalidad, la cual determina que se convierta en un *malware* o no. Este sistema de equipos es controlado desde la *botmaster* de forma remota y podrá ser usado para realizar actividades delictivas, a la par que interacciones normales para reducir el riesgo de ser detectados.

El problema viene dado por el hecho de que la mayoría de máquinas que forman parte de una *botnet* no se dan cuenta de ello, ya que suelen ser infectadas por un *malware*, por ejemplo, un gusano que las incorpora como *zombies* a la red. Es por esta razón que se ha decidido incluirlas en este estudio.

Los ataques manuales acaban siendo más costosos para los ciberdelincuentes, por lo que se tiende a optar cada vez más por estas opciones automáticas para casos como el de las redes sociales, consiguiendo controlar el contenido de varias maneras. Uno de los propósitos de un *bot* fraudulento es mejorar su reputación, no ser detectado y tener más información y acceso a cuentas reales, es por ello que engaña a las personas para conectar en la red social e imitar el comportamiento de esos perfiles.

Asimismo, las cuentas con las que haya conectado serán víctimas de todo el contenido compartido por el *bot*, incluso si es *spam*. Cabe destacar que el contenido que manipula un *bot* no suele ser aleatorio, sino que busca el que sea de interés respecto a un tema, con el fin de abrumar las publicaciones relacionadas con este, entorpeciendo la participación de los humanos en las discusiones que se estén llevando a cabo en ellas. También existen casos en los que el *bot* no solo pone comentarios, sino que también sube publicaciones a su perfil relacionadas con dicho tema, pero dirigidas a su interés y así contaminar los datos de búsqueda e incluso las opiniones que un humano pueda tener al respecto. [19]

Los usos más comunes de las *botnets* son los fines políticos o sociales, para apoyarlos o criticarlos y así hacer viral o tumbar una campaña específica. También se utilizan para realizar ataques DDoS y que las víctimas no puedan acceder a ellos, o para comprometer información como las cuentas y usarlas para hacer llegar campañas de *phishing* en su nombre, lo cual les aporta credibilidad, al hacer uso de una cuenta legítima.

Además, es muy accesible, ya que hay webs abiertas que ofrecen *botnets* a un precio bastante reducido, con la finalidad de subir los seguidores de cuentas en redes sociales e incluso para hacerse pasar por ciudadanos reales y hablar por las redes sociales sobre temáticas que apoyen, o no, un determinado discurso. [20]

Estas son las *botnets* sociales, cuya aparición viene dada por la influencia diaria que tienen las redes sociales en la sociedad, cambiando incluso la forma de actuar o pensar de las personas, la llamada influencia digital. El uso de estas es muy común y su existencia está confirmada en varios estudios que se han realizado y se pueden ver en: [21] [22] [23]

Es importante destacar que, aunque las *botnets* sociales solo han recibido atención recientemente, al verse demostrada su eficacia, es vital encontrar las contramedidas correspondientes a este tipo de ataques. Además, esto ya ha sido estudiado y se explicará a continuación, en unos sencillos pasos.

En primer lugar, mediante un ejemplo, se podrá demostrar la eficacia y las ventajas de explotar una *botnet* social para la distribución de *spam* en Twitter. Se ha escogido esta red social por ser una de las más conocidas y usadas en todo el mundo y este ejemplo de ataque viene motivado a que actualmente en esta red social solo se suspenden las cuentas que crean *tweets* de *spam* y no las que los retuitean, es decir, quienes los republican.

Para ello, se propone la optimización multiobjetivo, haciendo uso de un conjunto de *bots* sociales formando un árbol. Con esto se consigue minimizar el tiempo que tarda un *tweet* de *spam* en llegar a un número máximo de usuarios de Twitter; ya que, al retuitear siguiendo esa estrategia, se tienen más víctimas a menor costo del *botmaster*, siendo este el perfil que publica

el *tweet* de *spam* original que será retuiteado por el resto. Como la optimización es NP-hard, la solución es heurística y eficaz en Twitter y simulaciones controladas por trazas, es decir, la más satisfactoria.

En segundo lugar, es sencillo demostrar que una *botnet* social puede manipular con facilidad la influencia digital de los usuarios de Twitter porque, si colaboran para manipular las interacciones de sus usuarios objetivos, ya están siendo manipulados. Hay herramientas, como Klout, Kred y Retweet Rank, que miden dicha influencia en un usuario basándose exclusivamente en sus interacciones con otros. Este ataque puede ser eficaz en la orientación de anuncios, la mejora del servicio del cliente, la contratación y muchas otras aplicaciones.

Finalmente, en el artículo [24], en el cual me he basado para esta explicación, se proponen dos contramedidas para defenderse de estos dos ataques mencionados. La primera tiene que ver con la distribución de *spam,* en la que se propone mantener una puntuación de *spam* por usuario y que se actualice cada vez que se retuitee algo considerado como tal, suspendiendo al usuario si dicha puntuación supera un umbral predefinido. La segunda contramedida va dirigida al ataque que se encarga de manipular la influencia digital, tratando de defenderse encontrando suficientes usuarios creíbles y utilizar únicamente las interacciones originadas por estos para medir la influencia digital. Además, se ha confirmado la efectividad de ambas defensas a través de experimentos exhaustivos y estudios de simulación detallados, impulsados por conjuntos de datos del mundo real.

3. Análisis

En este apartado se realizará un análisis de dos redes sociales bastante dispares, como lo son Twitter e Instagram, con el fin de valorar la realización de una *botnet* social en una de ellas.

3.1 API

Para comenzar, se analizarán las API, ya que los recursos se obtienen a partir de estas y cada una los ofrece de manera distinta, la cual debe ser estudiada.

3.1.1 Twitter

Twitter es una red social de actualidad y en estos últimos años está entre una de las más usadas. Se caracteriza por la brevedad de sus publicaciones, restringiendo el número de caracteres con un máximo de 280, facilitando así la inmediatez a la hora de publicar un *tweet* y divulgando las noticias de una manera mucho más rápida, ya que permite retuitearlo, es decir, republicarlo en otra cuenta con un simple *click*, aumentando la visibilidad de aquello que se haya dicho en él. Además, esta red social dispone de una plataforma para desarrolladores y será a partir de la cual se basará todo este análisis sobre ella. [25]

En esta plataforma existen tres tipos de acceso: estándar, premium y de empresa, cada uno con más permisos que el anterior y siendo los dos últimos de pago. Para todas ellas, es necesario tener una cuenta en la red social y que esta sea cuenta de desarrollador, porque se necesitan las claves de la API. Para solicitar el acceso como desarrollador, se pasará por un proceso de aceptación bastante riguroso, el cual demora unos días, y Twitter se pone en contacto con la persona solicitante para que le aclare todo aquello que van necesitando.

En este caso, fue requisito esencial especificar que el acceso se requiere meramente para probar y analizar las funcionalidades que proporciona la API, y también será necesario aclarar para qué no se va a usar. Es decir, se tuvo que aclarar que con dicho acceso no se pretende realizar ningún análisis de *tweets*, usuarios o cualquier otro tipo de contenido, ni se tuiteará, retuiteará o se dará me gusta, ni se mostrará el contenido fuera de su plataforma; ya que Twitter lo considera como propósito comercial y no lo permite. Todo esto teniendo en cuenta que para este análisis no se pretende realizar ninguna inversión económica.

Una vez acepten el acceso, se puede crear una aplicación en el perfil de la plataforma para desarrolladores, que debe estar vinculada a un proyecto propio del mismo portal, para poder obtener las claves necesarias de acceso a la API: consumer_key, *consumer_secret*, *access_token*, *token_secret* y *bearer_token.*

Para este análisis, todas las pruebas han sido realizadas con Postman, una herramienta que permite realizar solicitudes a las API mediante una interfaz gráfica. Para ello, se ha seguido el tutorial específico de pruebas con Postman, siendo este el más específico y completo de entre los que ofrece Twitter en su portal para desarrolladores, ya que proporciona tres colecciones que pueden añadirse directamente a la herramienta: la API de Twitter en su versión 2, la API de anuncios de Twitter y los puntos de acceso de Labs. En general, la API de Twitter incluye una gran variedad de puntos de conexión, divididos en los siguientes cinco grupos principales: [26]

- Cuentas y usuarios. Sirven para programar la administración de un perfil o la configuración de alguna cuenta autorizada.
- Tweets y respuestas. Facilitan la muestra de *tweets* de cuentas específicas y la búsqueda de palabras clave en ellos. Un ejemplo de uso es la identificación de información errónea sobre iniciativas públicas de salud.
- Mensajes directos. Aclaran que no son vendidos, ya que esto es la parte privada de la red social, pero sí brindan el acceso a las conversaciones de los usuarios que hayan otorgado dicho permiso de manera explícita y, además, es posible crear experiencias de diálogos, como *chatbots*, que permiten a las empresas u otras organizaciones ofrecer una atención más inmediata.
- Anuncios. Ayudan a las empresas a crear, ejecutar y administrar campañas publicitarias.
- Herramientas y SDK de editor. Permiten integrar funciones de Twitter en una página web, por ejemplo, la función de compartir.

Mediante las pruebas realizadas se ha podido comprobar que el acceso está bastante más restringido de lo esperado respecto a lo que se detalla en lo que se ofrece para los desarrolladores, o al menos en la versión gratuita con la que se ha analizado. De forma predeterminada solo se ha conseguido acceder a información que ya es pública desde la propia red social. Aunque solo ha sido posible la obtención de información básica como el listado de *tweets* de un usuario, su identificador o el listado de seguidores; pero no ha sido posible realizar acciones que modifiquen algo en la red social, es decir, en las funcionalidades básicas como indicar que algo te gusta o retuitearlo, seguir o dejar de seguir una cuenta y bloquear o desbloquearla. Lo que devuelve la API para esas acciones es una respuesta con estado 403 y un mensaje de “Forbidden”, indicando que está prohibido el acceso.

Para tener más permisos se han de solicitar y, por consecuente, pagarlos. Con lo que se puede deducir que todas aquellas organizaciones que hagan uso de alguna de esas funcionalidades de forma automática y que, por lo tanto, necesitan dichos accesos de la API, están pagando para ello.

Una parte positiva respecto a la seguridad de la API es el control que tiene Twitter sobre quien realiza las solicitudes, ya que dependen de las cinco claves introducidas, por lo que además de restringir el acceso dependiendo de los permisos que tenga dicha cuenta de usuario a la que pertenecen las claves, también podrá ver desde qué cuenta se están realizando las acciones inadecuadas. El problema podría venir a la hora de identificar realmente a qué persona pertenece dicha cuenta de Twitter o si las claves han sido robadas a ese usuario y en realidad están siendo usadas por otra persona.

Por último, para esta primera red social cabe destacar que, aunque solo se haya realizado pruebas con Postman, se ha visto que hay una librería en el lenguaje de programación Python que facilita el acceso a la API de Twitter, llamada Tweepy [27], a pesar de que no se incluya en los tutoriales que ofrece esta red social en su plataforma para desarrolladores.

3.1.2 Instagram

La otra red social escogida ha sido Instagram, para que fuesen dos redes sociales bastante distintas de entre las más usadas y conocidas actualmente. La diferencia más evidente es que Instagram destaca por su contenido mayormente visual, mediante imágenes y vídeos, y menos textual, siendo publicaciones que suelen ser más elaboradas y menos instantáneas que las de Twitter, en general. Aunque cabe destacar que después surgieron las "historias", con el fin de que fuesen más inmediatas que la publicación habitual, por el hecho de borrarse a las 24 horas de haberse subido, sin afectar así al *feed* de perfil, es decir, al tablero de publicaciones subidas por una cuenta, en el que se visualizan todas en orden cronológico. Además de este cambio, esta red social en los últimos años ha sido una de las que más cambios y más crecimiento ha experimentado, en parte a raíz de que fuese comprada por Facebook el 9 de abril de 2012, ofreciendo cada vez más funcionalidades, tipo de publicaciones, etc.

Instagram divide su API en dos: "API Graph" y "API de visualización básica". La primera está basada en la API Graph de Facebook y está diseñada para las cuentas verificadas por esta red social como creadores de contenido o empresas, que deben pasar por ese proceso previo de verificación, ya que a estas se les aportan datos adicionales, como las estadísticas y control más completo de todas sus interacciones en los medios sociales, porque su trabajo o parte de este se basa en ello y, por lo tanto, lo necesitan. Todo ello con la limitación de la ordenación de los resultados o la admisión de los *reels*, un tipo de publicación que se basa en ser vídeos cortos que no se puede avanzar ni retroceder en ellos. Sin embargo, la otra API está enfocada al usuario común, por lo que solo ofrece la obtención de información más básica del perfil y las publicaciones de sus cuentas en la red social; aunque desde ella se puede realizar la lectura de datos básicos de otra cuenta si son datos que esa cuenta pueda ver en la aplicación, a excepción de los archivos IGTV, historias, *reels* y comentarios, que no se obtienen en ningún caso. [28]

Las pruebas en este caso, en vez de ser realizadas desde Postman, se han hecho a partir de código escrito en lenguaje Python, ya que este cuenta con un módulo llamado "Instabot" que además de implementar la envoltura sobre la API de Instagram, también ofrece otras funciones como: seguir a una lista de personas; hacer *like*, es decir, indicar que te gustan, unas publicaciones según un hashtag; o dejar de seguir a las personas que no te siguen; entre otras. Todo ello lo hace un módulo muy completo e interesante de utilizar si se quieren hacer pruebas mediante código sobre esta red social. Además, a pesar de ser relativamente nuevo, se actualiza muy a menudo, mejorando cada vez más, ya que su desarrollo está activo.

Instabot ofrece documentación para desarrolladores, proporcionando la explicación de los métodos del *bot* como si fuese otra API. Un detalle de este módulo es que se esconde bajo la máscara del teléfono móvil para que los servidores de Instagram piensen que sus solicitudes se realizan desde alguna aplicación móvil y no desde un código. De todos modos, ha sido descubierto varias veces por la red social, así que cada vez intentan mejorar en este aspecto, para que acabe pasando desapercibido.

Por ejemplo, la API de Instagram restringe la cantidad de llamadas que pueden realizarse en un cierto tiempo, razón por la cual hay que configurar los *bots* para saltarse esas restricciones, esperando un cierto tiempo cada cierto número de acciones para volver a ejecutar y así sucesivamente, y esto es lo que hace Instabot. Además, incluye otras autolimitaciones, como un número máximo de *likes* a una publicación, actualmente marcado en 999. Todo esto lo limita además de hacer que los procesos se demoren más, pero evita llamar la atención, siendo esto lo que se pretende conseguir.

De todos modos, no se precisa de la solicitud de una cuenta de desarrollador, como ocurre en Twitter y esto es lo que ha llevado a que las pruebas se hayan realizado desde una cuenta ficticia, para evitar poner en el *bot* la contraseña de la cuenta personal. Por lo que tiene su parte negativa y positiva a la vez ya que, a pesar de todas las limitaciones comentadas, en general, mediante la API, aunque sea indirectamente, se pueden realizar más operaciones que en la de Twitter, según el acceso común desde el que se han realizado las pruebas en ambos casos. Pero Instagram, respecto al método de autenticación para poder realizar dichas pruebas, tiene menos en cuenta la privacidad del usuario, porque solicita en el código tanto usuario como la contraseña con el que se puede iniciar sesión en dicha cuenta y realizar acciones desde la misma, pudiendo ser o no la persona propietaria; algo que no se puede conseguir mediante las claves que se introducen para probar la API de Twitter. Todo esto, teniendo en cuenta que me limito a las pruebas realizadas.

Como parte positiva respecto a la seguridad, en este caso también sería el control que puede tener Instagram sobre quién realiza las solicitudes, aunque se haga uso del *bot*, ya que en este se

debe iniciar sesión con los datos de inicio de sesión reales de una cuenta, tal y como se ha comentado, antes de realizar cualquier otra acción; pudiendo saber desde qué cuenta se están realizando las acciones inadecuadas. El problema sigue siendo a la hora de identificar realmente a qué persona pertenece dicha cuenta de Instagram o si los datos de inicio de sesión han sido robados a ese usuario y en realidad están siendo usadas por otra persona.

3.2 Técnicas de ataque

Este apartado está focalizado desde el punto de vista de un ciberatacante, es decir, se expondrán algunas técnicas de ataque y se explicarán si pueden ser utilizadas o no sobre las dos redes sociales escogidas y de qué manera, con el fin también de realizar una comparación entre estas. Cabe aclarar que el objetivo no es realizar un ataque a la red social en sí, sino a los consumidores de la misma.

3.2.1 Enlaces maliciosos

Uno de los métodos más comunes para la propagación de *malware* es mediante enlaces maliciosos que a priori transmita confianza al usuario y además sienta la necesidad de acceder a él, pero en realidad te instale un *malware* en el dispositivo o lleve a un sitio web falso que imite una web oficial legítima, con el fin de que introduzca datos personales de inicio de sesión e incluso bancarios.

Estos enlaces pueden enviarse por distintos medios, pero en este análisis se va a especificar cómo podrían ser mandados o no por las redes sociales escogidas. Partiendo de la base de que un enlace puede escribirse en cualquier texto, se podrá compartir en cualquier parte de las redes sociales en las que se pueda escribir; pero lo interesante de un enlace no es eso, sino que el acceso sea directamente desde el sitio en el que se encuentre y no que la persona que lo vea deba ir a un navegador a copiar a mano la dirección, ya que esto llevará a que finalmente nadie haga esta labor tan pesada, sobre todo en enlaces largos, porque cada vez nos acostumbramos a tenerlo todo de manera más fácil y accesible.

Ahora bien, esa accesibilidad se consigue cuando directamente pinchando sobre el enlace lleve a aquello que teóricamente indica y en Twitter lo tienen muy bien integrado, ya que en sus publicaciones, es decir, en los *tweets*, si se escribe o se copia un enlace, lo identifica como tal, haciéndolo accionable y, por lo tanto, accesible por todas las personas que lo vean. Además, si alguien lo retuitea, ese *retweet*, al ser idéntico, también lo llevará y, de esa forma, se puede difundir el enlace de una manera muy rápida. Esto conllevará a que, aunque sea publicado por una cuenta que el usuario no conoce, es posible que pueda verlo en un perfil de algún contacto que tenga, lo cual hará que sea más fiable y aumente la probabilidad de que caiga en la trampa.

Sin embargo, en ninguno de los tipos de publicaciones con los que cuenta Instagram existe esa accesibilidad, ya que el texto no reconoce los enlaces y no los pone como tal, sino como texto plano, que además ni se puede seleccionar para poder copiar de manera automática. Esto lo hace más difícil para el usuario, el cual opta por ni siquiera buscarlo en un navegador, al tenerlo que copiar a mano, a no ser que esté extremadamente interesado por aquello que indique que es, pero eso no suele ocurrir, por lo que no interesa desde el punto de vista del atacante.

De todos modos, sí hay otras maneras de difundir un enlace de la forma correcta, pero es bastante más complejo, ya que es mediante una historia de Instagram en la que se incluya específicamente el enlace al igual que se añade un *gif*, pero esta función es nueva y no aparece en todas las cuentas todavía. Lo que sí existe desde hace más tiempo es el *swipe up* en las cuentas que tienen más de 10 mil seguidores. Este requisito es para que solo aquellas cuentas más influyentes puedan compartir de manera rápida un enlace a alguna prenda u otra cosa que promocionen en su historia, simplemente deslizando el dedo de abajo a arriba de la pantalla; lo cual sigue siendo igual de cómodo, pero tampoco es posible realizarlo desde un *bot* que seguramente acaba de empezar a tener seguidores, así que tampoco es tan sencillo atacar de dicha manera. Esta misma opción se ofrece en los anuncios que se pueden comprar a Instagram, pero es algo que tampoco interesa, ya que se debe pagar para ello y pasar por varios controles y condiciones para que el anuncio sea publicado.

En ambas redes sociales, tanto en la descripción del perfil como en los mensajes directos, se pueden compartir los enlaces con su vínculo asignado, pero sigue sin ser una forma viable si lo que se quiere es que llegue al mayor número de usuarios posibles. En el caso de la descripción del perfil se puede poner un solo enlace y también se tendría que conseguir el paso previo de que entren a ver el perfil e incentivar para que pinchen en dicho enlace, el cual suele pasar desapercibido. Respecto a los mensajes directos, al ser algo más personal, los usuarios no suelen confiar en lo que le mandan cuentas desconocidas, bien sea a la conversación individual o a algún grupo que haya creado con más cuentas que no conoces, o que al entrar a ver su perfil no parezcan fiables, siendo esta revisión lo más común al ver el mensaje inesperado.

Por lo tanto, en general, Instagram protege más a sus usuarios de la propagación de *malware* de esta manera, haciéndola más segura; aunque "incómoda" para aquellas personas que sí quieran compartir un enlace benigno. Pero esta consecuencia se debe asumir con el fin de priorizar la seguridad ante la comodidad. Además, suele ser muy común el hecho de que aportar seguridad a un sistema lo haga más incómodo y/o engorroso para los usuarios, como la autenticación en dos pasos, lo cual demora más el proceso de acceder a un sitio, pero lo hace más seguro y cada vez más personas lo utilizan en los sistemas que los ofrecen, así que esto se debería entender de la misma manera.

3.2.2 Ingeniería social

La ingeniería social es la técnica que consiste en obtener de usuarios legítimos su información confidencial, acceso o permisos de sistemas a través de la manipulación, es decir, sin que estas personas estén al corriente. De esta manera, con dichos accesos y/o permisos es posible ocasionar daños a otras personas u organismos que estén comprometidos, a partir del primero que ha sido víctima. El principio que sustenta esta práctica es que los usuarios son el eslabón más débil de cualquier sistema, lo cual ha sido comprobado en muchas ocasiones.

Dicha manipulación se puede realizar de varias maneras y es por ello que hay muchos ejemplos de ingeniería social, por ejemplo, un punto débil de las personas suele ser sus ganas de ser popular o también su familia, en muchas ocasiones su pareja o similar, en caso de tenerla, y/o los celos que pueda sentir por ella. Relacionando esto con las redes sociales, siendo estas un lugar en el que las personas se relacionan y pueden enviarse mensajes, vídeos y fotos, es sencillo de plantear un ciberataque.

Algo muy común son las aplicaciones o sitios web que se aprovechan de dichas vulnerabilidades para ofrecer algo que lo solucione, como aumentar los seguidores o averiguar qué publicaciones le gustan a otra cuenta. Esto lo hace llamativo para las personas que quieran conseguir esa información y/o esos seguidores, pero para ello se necesita entrar con las credenciales de Instagram. Lo que ocurre es que, en cuanto el usuario las introduzca, el ciberdelincuente que haya realizado ese proceso de ingeniería social se las guardará con el fin de acceder posteriormente y realizar acciones fraudulentas; pudiendo ocurrir tanto en Instagram como en Twitter a partes iguales, e incluso en cualquier red social.

3.2.3 Esteganografía

La esteganografía es la práctica de ocultar información secreta e invisible en algo que no lo sea, es decir, en aquello que aparentemente sea inocente, visible y suele pasar desapercibido. En este caso, hablamos específicamente de la esteganografía digital, es decir, la que lo oculta en archivos multimedia que podrían pasar por cualquier canal sin levantar sospechas.

Esto tiene varios usos, bien sea el de ocultar directamente el código malicioso, pudiendo ser instrucciones a seguir en un ataque determinado, por ejemplo, como texto dentro de una imagen; o que sea la forma de comunicarse de los *bots* de una *botnet* social. El *bot* podría introducir el mensaje de manera oculta en la imagen, y otro podría leerlo y contestarle de la misma manera, evitando así ser descubiertos a simple vista, ya que se estarían mandando publicaciones con imágenes, algo común en las redes sociales, e incluso sin la necesidad de mandarlo directamente a esa cuenta objetivo, lo cual debería hacerse si contiene el código malicioso, sino bastaría con publicarlo en la cuenta de manera pública.

De esta forma, es todavía más difícil que sea descubierta, porque no se encontraría la relación entre los *bots*. Aunque la red social podría revisar las imágenes que estén compartidas en su red social en busca de información oculta, mediante el estegoanálisis, que consiste en analizar computacionalmente unos contenidos sospechosos para determinar presencia de algún tipo de desviación estadística respecto a sus análogos inocuos. En el caso de que se hallase alguna anomalía, se revisaría si concuerda con alguna técnica esteganográfica concreta, pero a pesar de ser algo automático, esto no se hace hasta el momento, sino que comprimen las imágenes y de esa manera la información oculta puede ser borrada, si esto no se tiene en cuenta.

Para que eso ocurra, lo que se debe hacer es averiguar las características que tendrá una foto después de ser subida a la red social e imitarlo en la que lleve la información oculta y así evitar que realice cambios en ella al comprimirse.

Por lo tanto, después de la explicación, se podría decir que es una técnica que puede ser usada en prácticamente cualquier red social o por lo menos sí que se podría tanto en Twitter como en Instagram, ya que es común la compartición de imágenes, con sus características correspondientes.

3.2.4 Web scraping

Hacer web *scraping* no es ilegal, sino alegal. En concreto, es un proceso dentro de *Data Science* que es utilizado para extraer y almacenar datos de sitios web normalmente públicos, simulando la navegación de un ser humano por ella y evitando que se sepa que es un programa informático. Esto se consigue haciendo uso de los datos obtenidos por peticiones HTTP para el entrenamiento y después analizarlo mediante *machine learning*, en cuyo código se suele usar el lenguaje de programación Python que, aunque no es imprescindible, se ha visto que es lo más común actualmente.

Como se debe programar la forma en la que el *bot* va a navegar por el sitio web, dicho sitio debe ser conocido para poder indicar algunas pautas y ahí es donde se produce el problema principal de esta técnica ya que, cuando cambia la interfaz de la web, dichas indicaciones podrían dejar de ser válidas. Por lo tanto, este no es un método infalible, pero sí una forma de funcionar, que se debe ir revisando y actualizando periódicamente.

Como Twitter e Instagram tienen páginas web, esta técnica podría ser usada. Aunque en el caso de Instagram, todavía no ofrece en su sitio web todo lo que ofrece en la aplicación, ya que esta surgió primero, además de todos los cambios que esta red social sufre muy a menudo, lo cual haría cambiar el código frecuentemente. En el caso de Twitter, sería más sencillo, ya que se producen menos cambios y además el sitio web ofrece todas las funcionalidades de la misma manera que la aplicación.

3.2.5 Ataque de fuerza bruta o de diccionario

Un ataque mediante fuerza bruta se refiere al acceso a una cuenta ajena, mediante la comprobación de todas las combinaciones posibles hasta encontrar la correcta. Este no suele utilizarse como primera opción, ya que la gran cantidad de combinaciones harán que el proceso se demore demasiado, siendo exponencial según la cantidad de caracteres que tenga la contraseña, dato que tampoco se suele saber con anterioridad. Es por ello, que es más común el uso del ataque de diccionario, ya que reduce los intentos centrándolo en las palabras del diccionario con el que se vaya a probar, pudiendo ser el de una lengua en específico o diccionarios creados con ese fin y que incluyen contraseñas comunes.

En cualquier caso, tanto en Instagram como en Twitter, sí se tiene precaución en este aspecto, ya que la autenticación es la parte en la que más se suele integrar y aplicar la seguridad en prácticamente cualquier plataforma. En este caso, la autenticación está restringida a un cierto número de intentos e incluso avisa mediante el correo electrónico que se tenga vinculado, en el caso de que se detecte algún intento inusual de inicio de sesión, además de avisar cuando se inicia sesión en un dispositivo en el que no estaba la cuenta abierta, con el fin de que la persona propietaria de dicha cuenta pueda corroborar que ha sido ella. Además, cada vez tiene más auge el uso de un segundo factor de autenticación.

4. Diseño

Este proyecto constará de una API flexible, lo cual permite usar cualquier lenguaje de programación o herramienta por parte del cliente, de manera independiente al que se decida emplear para programarla, mejorando su usabilidad y fomentando su uso. Esta ofrece funcionalidades que aportarán valor a la información que nos proporcionan las redes sociales de manera pública, aunque haya que crearse una cuenta para ello. Con esta se podrán detectar usos inapropiados de las redes sociales, como por ejemplo la presencia de *botnets* maliciosas, noticias falsas, entre otros fraudes.

Asimismo, se favorecerá la integración de aspectos tanto de *big data* como de *machine learning*, pudiendo exfiltrar información de forma programática para ello, permitiéndose recolectar los macrodatos con multitud de aplicaciones y, de esta manera, poder actuar en consecuencia de todo lo detectado mediante la información conseguida.

4.1 Funcionalidades planteadas

El proyecto contará con las siguientes funcionalidades que se han valorado como las más propensas o con las que de mejor manera se conseguirá la detección del uso de las *botnets* sociales y otras acciones fraudulentas en las redes sociales. Se pretende que esta API sea lo más flexible posible y que no vaya enfocada a solo una red social, por lo que se podrán hacer consultas especificando sobre cuál se quiere obtener la información que se solicite, lo cual se podrá decidir en cada una de las llamadas.

En primer lugar, se especifican las funcionalidades que devuelven información a partir del estado actual de las redes sociales, es decir, que se extraerá de ellas directamente y serán divididas entre los distintos aspectos que podemos encontrar en las redes sociales y, por último, se expondrán las funcionalidades relacionadas con los datos de auditoría propios de la API.

4.1.1 Cuentas

Las cuentas se refieren a los usuarios registrados en una red social, para poder interactuar en ella y tienen su respectivo perfil.

4.1.1.1 Contenido

Esta funcionalidad se basa en devolver todo el contenido multimedia y/o escrito que haya sido compartido desde una cuenta, sin ser comprimido ni manipulado, para poder ser analizado posteriormente a su obtención. Además del momento en el que se creó la cuenta, ya que también se ha valorado como un dato de interés sobre las mismas.

4.1.1.2 Relaciones

Las relaciones entre cuentas son el hecho de que un usuario siga, sea seguido, le guste o interaccione con el contenido de otro, creando así una relación entre ellos.

La funcionalidad que se propone respecto a esto se basa en devolver la lista de las cuentas que siguen y las que son seguidas por el usuario que se indique en la petición, además del listado de las que han indicado que les gusta algún contenido de dicho usuario y las que hayan realizado alguna otra interacción con él.

El objetivo es poder realizar la misma llamada con cada una de esas cuentas listadas y así encontrar relaciones sospechosas entre estas, pudiendo ser incluso circular en algún caso si las propias *botnets* sociales se siguen entre ellas o siguen a las mismas cuentas; por ejemplo, siendo estas las que quieren manipular.

4.1.1.3 Ranking

Esta funcionalidad se basa en devolver las X cuentas más y/o menos populares en las redes sociales indicadas, siendo X un número indicado en la llamada.

La popularidad puede valorarse de varias maneras, por lo que en este caso habrá tres opciones de consulta: la primera se basa en la cantidad de seguidores que tengan; la segunda, en la cantidad de *likes* recibidos por las cuentas, siendo estos en una red social lo que comúnmente se refiere al hecho de que un usuario indique que le gusta una publicación; y la tercera, en el resto de interacciones que pueden recibir las cuentas directa o indirectamente, posiblemente mediante sus publicaciones.

En el caso de que haya empates, el orden será el indicado en la llamada, pero si dicho valor es nulo o inadecuado, por defecto se priorizarán las últimas cuentas creadas, ya que es más probable que las *botnets* sociales se encuentren entre esas, es decir, entre las menos longevas de la red social. Los valores que se considerarían como válidos serían: DESC y ASC. De manera que el primero es el que está por defecto y el segundo lo contrario, es decir, en el orden cronológico que se crearon.

Además, se pueden añadir filtros con condiciones relacionadas con cualquiera de los datos que se puedan obtener de un comentario. Esto se incluye con el fin de reducir la búsqueda en alguno de los casos que se esté estudiando.

Tipo	**Llamada**	**Cuerpo**	**Respuesta**

GET	/user/content	{
		user: <string>,
		media: <bool>,
		text: <bool>
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			media: <array<bytes>>
			text: <array<string>>,
			date: <datetime>
			},
			instagram: {
			media: <array<bytes>>
			text: <array<string>>
			date: <datetime>
			}
			}

GET	/user/relationship	{
		user: <string>
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			followers: <array<string>>,
			following: <array<string>>,
			likes: <array<string>,
			interactions: <array<string>>
			},
			instagram: {
			followers: <array<string>>,
			following: <array<string>>,
			likes: <array<string>,
			interactions: <array<string>>
			}
			}

GET	/user/ranking/follow	{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			user: <string>,
			followers: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			followers: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			user: <string>,
			followers: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			followers: <int>
			},
			…
			}
			}
			}

GET	/user/ranking/like	{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			user: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			likes: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			user: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			likes: <int>
			},
			…
			}
			}
			}

GET	/user/ranking/
	interaction
		{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			user: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			interactions: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			user: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			user: <string>,
			interactions: <int>
			},
			…
			}
			}
			}

Tabla 1. Especificación de llamadas relativas a las cuentas

4.1.2 Publicaciones

Las publicaciones son las distintas formas que tiene el usuario de compartir el contenido desde su perfil, pudiendo ser o no variado dependiendo de la red social.

4.1.2.1 Estadísticas

Esta funcionalidad se basa en devolver los datos estadísticos de la frecuencia con la que un usuario sube publicaciones en las redes sociales indicadas según la línea temporal.

Más específicamente, se han escogido los datos que ofrecen información que se puede considerar de utilidad, siendo estos: el mínimo, el máximo, la media, la mediana, la moda, la varianza, la desviación típica y el coeficiente de variación.

4.1.2.2 Ranking

Esta funcionalidad se basa en devolver las X publicaciones que más y/o menos populares sean en las redes sociales indicadas, siendo X un número indicado en la consulta.

La popularidad puede valorarse de varias maneras, por lo que en este caso habrá dos opciones de consulta: la primera se basa en la cantidad de *likes* que tenga la publicación y la segunda, en la cantidad del resto de interacciones que haya recibido, pudiendo variar en función de cada red social. Por ejemplo, en Twitter se referiría a los retuits y en Instagram puede ser las veces que se haya guardado o compartido la publicación.

En el caso de que haya empates, el orden será el indicado en la llamada, pero si dicho valor es nulo o inadecuado, por defecto se priorizarán las últimas publicaciones que se hayan subido a la red social, siendo estas más recientes y, por lo tanto, es más probable que pertenezcan a una cuenta activa, que son las que más interesan en este caso. Ocurriría lo mismo en el caso de poner “DESC”, pero si se introduce “ASC” el orden sería el opuesto.

Además, se pueden añadir filtros con condiciones relacionadas con cualquiera de los datos que se pueden obtener de una publicación. Esto se incluye con el fin de reducir la búsqueda en alguno de los casos que se esté estudiando.

4.1.2.3 Coincidencias

Esta funcionalidad se basa en devolver el listado de identificadores de publicaciones que contengan, en su texto relacionado, la o las palabras clave indicadas en la consulta.

4.1.2.4 Contenido

Esta funcionalidad se basa en devolver todos los datos de una publicación a partir de su cadena de identificación. De esta manera, se pueden ver los datos de cada una de las publicaciones que, por ejemplo, sean devueltas en otras llamadas.

Tipo	Llamada	Cuerpo	Respuesta

GET	/post/content	{
		post: <string>
		}
			{
			code: <int>,
			message: <string>,
			post: {
			socialmedia: <string>,
			user: <string>,
			date: <datetime>,
			media: <array<bytes>>,
			text: <string>,
			comments: <array<string>>,
			likes: <array<string>>,
			interactions: <array<string>>
			}
			}

GET	/post/coincidence	{
		words: <string>,
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			posts: <array<string>>
			},
			instagram: {
			posts: <array<string>>
			}
			}

GET	/post/statistic	{
		user: <string>,
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			minimum: <decimal>,
			maximum: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			},
			instagram: {
			minimum: <decimal>,
			maximum: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			}
			}

GET	/post/ranking/like	{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			post: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			post: <string>,
			likes: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			post: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			post: <string>,
			likes: <int>
			},
			…
			}
			}
			}

GET	/post/ranking/
	interaction
		{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			post: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			post: <string>,
			interactions: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			post: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			post: <string>,
			interactions: <int>
			},
			…
			}
			}
			}

Tabla 2. Especificación de llamadas relativas a las publicaciones

4.1.3 Comentarios

Un usuario puede poner un comentario como respuesta a cualquier publicación subida a una red social, siempre y cuando esta sea de una cuenta pública o, en caso de ser privada, el comentador sea seguidor de dicha cuenta y, por lo tanto, tenga acceso a sus publicaciones. Con estos comentarios son con los que se suele influenciar más respecto a la opinión sobre algo publicado o mediante *spam* se consigue que no se lean los comentarios reales y así que lleguen al menor número de personas posible.

4.1.3.1 Estadísticas

Esta funcionalidad se basa en devolver los datos estadísticos de frecuencia, en este caso, dividida en dos. Una será respecto a la frecuencia con la que un usuario comenta en las redes sociales indicadas según la línea temporal, independientemente de la publicación en la que se haya realizado dicho comentario; y la otra, respecto a la frecuencia de comentarios que tiene una publicación en específico, independientemente de la cuenta que haya realizado cada uno de esos comentarios.

Más específicamente, se han escogido los datos que ofrecen información que se puede considerar de utilidad, siendo estos: el mínimo, el máximo, la media, la mediana, la moda, la varianza, la desviación típica y el coeficiente de variación.

4.1.3.2 Coincidencias

Esta funcionalidad se basa en devolver el listado de identificadores de comentarios que contengan, en su texto relacionado, la o las palabras clave indicadas en la consulta.

4.1.3.3 Ranking

Esta funcionalidad se basa en devolver los X comentarios que más y/o menos populares sean en las redes sociales indicadas, siendo X un número indicado en la consulta. La popularidad puede valorarse de varias maneras, por lo que en este caso habrá dos opciones de consulta: la primera se basa en la cantidad de *likes* que tenga la publicación y la segunda, en la cantidad del resto de interacciones que haya recibido, pudiendo variar en función de cada red social. Por ejemplo, en Instagram se refiere a que el comentario reciba una respuesta y en Twitter además de esta, también contarían los retuits que dicho comentario recibe.

En el caso de que haya empates, el orden será el indicado en la llamada, pero si dicho valor es nulo o inadecuado, por defecto se priorizarán los últimos comentarios que se hayan realizado en la red social, por el mismo motivo que es así en las publicaciones. Ocurriría lo mismo en el caso de poner “DESC”, pero si se introduce “ASC” el orden sería el opuesto.

Además, se pueden añadir filtros con condiciones relacionadas con cualquiera de los datos que se pueden obtener de un comentario. Esto se incluye con el fin de reducir la búsqueda en alguno de los casos que se esté estudiando.

4.1.3.4 Contenido

Esta funcionalidad se basa en devolver todos los datos de un comentario a partir de su cadena de identificación. De esta manera, se pueden ver los datos de cada uno de los comentarios que, por ejemplo, sean devueltos en otras llamadas.

Tipo	**Llamada**	**Cuerpo**	**Respuesta**

GET	/comment/statistic/user	{
		user: <string>,
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			min: <decimal>,
			max: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			},
			instagram: {
			min: <decimal>,
			max: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			}
			}

GET	/comment/statistic/post	{
		post: <string>,
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			min: <decimal>,
			max: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			},
			instagram: {
			min: <decimal>,
			max: <decimal>,
			mean: <decimal>,
			median: <decimal>,
			mode: <decimal>,
			variance: <decimal>,
			deviation: <decimal>,
			variation: <decimal>
			}
			}

GET	/comment/coincidence	{
		words: <string>,
		twitter: <bool>,
		instagram: <bool>
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			comments:
			<array<string>>
			},
			instagram: {
			comments:
			<array<string>>
			}
			}

GET	/comment/ranking/like	{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			comment: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			comment: <string>,
			likes: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			comment: <string>,
			likes: <int>
			},
			…
			},
			min: {
			{
			comment: <string>,
			likes: <int>
			},
			…
			}
			}
			}

GET	/comment/ranking/
	interaction
		{
		amount: <int>,
		max: <bool>,
		min: <bool>,
		order: <string>,
		twitter: <bool>,
		instagram: <bool>,
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			twitter: {
			max: {
			{
			comment: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			comment: <string>,
			interactions: <int>
			},
			…
			}
			},
			instagram: {
			max: {
			{
			comment: <string>,
			interactions: <int>
			},
			…
			},
			min: {
			{
			comment: <string>,
			interactions: <int>
			},
			…
			}
			}
			}

GET	/comment/content	{
		comment: <string>
		}
			{
			code: <int>,
			message: <string>,
			comment: {
			socialmedia: <string>,
			user: <string>,
			date: <datetime>,
			media: <array<bytes>>,
			text: <string>,
			likes: <int>,
			interactions: <int>
			}
			}

Tabla 3. Especificación de llamadas relativas a los comentarios

4.1.4 Auditoría

Por motivos de seguridad, la API guardará una auditoría de todas las llamadas realizadas a la misma, más específicamente, los campos que se han introducido para realizarla y las respuestas obtenidas por ellas, además de los errores que hayan podido ocurrir en el proceso.

En este caso, la información no será obtenida de las redes sociales, sino de aquello que se ha guardado en la base de datos propia de la API, al realizarle llamadas, ya que en ellas estarán los datos de dicho momento, aunque no coincida con el estado actual de la red social.

4.1.4.1 Historial

El historial de consultas no es totalmente accesible por los clientes, sino que solo podrán obtener el listado de entradas, salidas y errores que hayan sido guardados a raíz de una llamada realizada desde el usuario de dicho cliente. De esta manera, se evitará que cualquier cliente de la API pueda ver el historial de otro, consiguiendo así la privacidad de los mismos. Este control deberá realizarse mediante los datos de sesión.

Como es posible que, al realizar la misma consulta, dependiendo del momento en que se realice, la respuesta pueda variar si dicha información ha sido alterada en la red social, obteniendo los datos de las anteriores se podrán hacer comparaciones que aportarán información añadida.

Asimismo, como en la auditoría se trabaja con un gran volumen de datos, se incluye un filtrado para encontrar la información de manera más concreta respecto a lo que se necesite o se esté estudiando.

Tipo	**Llamada**	**Cuerpo**	**Respuesta**

GET	/query/log	{
		filters: {
		{
		field: <string>,
		operator: <string>,
		condition: <string>
		},
		…
		}
		}
			{
			code: <int>,
			message: <string>,
			log: {
			{
			id: <string>,
			date: <datetime>,
			call: <string>,
			input: <json>,
			output: <json>,
			errors: <array<string>>
			},
			…
			}
			}

Tabla 4. Especificación de llamadas relativas a la auditoría

4.2 Limitaciones

A continuación, se incluyen las limitaciones que tendrá la API diseñada:

- Cada cliente de la API solo podrá realizar 3 consultas diarias a la misma de manera gratuita y, en el caso de que requiera hacer más, deberá pagar por el servicio.
- En las respuestas solo irá incluida la información que sea pública en la red social para cualquier usuario de la misma, pero esto no sería un impedimento para conseguir el objetivo propuesto, ya que generalmente las *botnets* utilizan cuentas pública y dicha actividad sí que sería devuelta por la API.
- La API no saca conclusiones de la información aportada, esa labor deberá ser de cada cliente, usándola y procesándola de la manera que considere más efectiva.
- Aunque sea posible especificar sobre qué red o redes sociales se quiere obtener la información de la llamada que se realiza, solo devolverá datos respecto a aquellas para las que se haya desarrollado, comenzando por Twitter e Instagram.
- Es posible que otra limitación sea el tiempo de respuesta de las llamadas, ya que es algo que no se puede medir sin realizar la implementación y las pruebas de rendimiento oportunas.
- Temporalmente, otra de las limitaciones que tiene la API es que solo sería posible la obtención de los datos relacionados con Twitter e Instagram, aunque actualmente existen muchas otras redes sociales.

5. Aplicabilidad

Una parte imprescindible de una idea es la aplicabilidad que esta pueda tener, así que se procede a explicar los distintos aspectos que se han considerado relevantes a la hora de llevar el proyecto a cabo.

5.1 Tecnologías

En este subapartado se detallarán las técnicas propuestas que serán necesarias para realizar el proyecto y el motivo por el que han sido las elegidas.

5.1.1 Lenguaje de programación

Para comenzar a desarrollar es importante saber en qué lenguaje será escrito el código, así que será lo primero que se especificará en este apartado. Para esta API se propone usar Python, ya que cuenta con gran cantidad de librerías y en concreto varias que son especialmente útiles para el uso de redes sociales; por ejemplo, en el caso de las escogidas para iniciar el proyecto, se cuenta con la librería Instabot para Instagram y con Tweepy para Twitter. Además, este lenguaje de programación facilita la integración con *machine learning* y es uno de los más usados para códigos enfocados a los ámbitos de la investigación, las matemáticas, la ciencia de datos y a la ciberseguridad.

Un motivo por el que se usa en ciberseguridad es porque Python incluye una librería para el uso de Metasploit, una herramienta para desarrollar y ejecutar *exploits* contra una máquina remota, lo cual permite realizar auditorías de seguridad, desarrollar y probar *exploits* propios, y probar las vulnerabilidades de sistemas informáticos bien sea para protegerlos o con fines de piratería informática. Siendo un *exploit* un programa informático o simplemente una secuencia de comandos que usa un agujero de seguridad o vulnerabilidad, aprovechándolos para provocar un comportamiento imprevisto o no intencionado en un *software*, *hardware* o cualquier dispositivo electrónico, normalmente con fines maliciosos como la instalación de *malware*, pero también puede utilizarse para protegerse de ellos, ya que no es un código malicioso en sí mismo, sino la llave para que estos accedan al sistema. [29]

Como punto negativo de que este lenguaje gradualmente se esté convirtiendo en uno de los principales en muchos ámbitos como los ya nombrados, destaca la gran diferencia e incompatibilidad que hay entre versiones, principalmente por la coexistencia y la invocación de múltiples versiones de módulos de Python.

De todos modos, al no obligar al cliente de la API a que use este mismo lenguaje, a priori no se cree que sea un problema grave en este caso en particular, al menos para los clientes de esta, sino que sería algo a tener en cuenta en su desarrollo; pero para ello existen soluciones. Una de ellas es el uso de varios entornos separados para cada tarea de desarrollo mediante Docker, herramienta para empaquetar aplicaciones y dependencias en contenedores ligeros, y otra solución es usar directamente "virtualenv", que es propio del lenguaje y sirve para crear entornos independientes de Python. [30]

En lo personal, no tenía experiencia previa con este lenguaje de programación, ya que no me ha sido necesario usarlo en ninguna de las asignaturas del grado ni del máster que he cursado, ni tampoco en mi vida laboral ni personal hasta el momento. Esto me ha hecho dudar en dicha elección, pero finalmente ha acabado siendo un incentivo, tomándolo como una oportunidad para aprender sobre Python y aumentar así mis conocimientos de programación.

5.1.2 Infraestructura

Prácticamente cualquier plataforma como servicio (PaaS) facilita mucho el construir, mantener y escalar las aplicaciones alojadas en ella, pero por el momento se ha escogido una de las más conocidas, Google App Engine, en parte porque es compatible con muchos lenguajes entre los que se encuentra Python.

Una de las mayores ventajas que tiene es que Google se encarga de todo lo relacionado con la infraestructura, facilitando así la escalabilidad y resiliencia de los proyectos alojados en dicha plataforma. En esta, la API se escala de forma automática en función del tráfico de uso, sin consumir recursos cuando el código no se está ejecutando. De esta manera, solo se paga por aquellos recursos que verdaderamente sean consumidos, evitando realizar estimaciones que puedan alejarse de la realidad y consiguiendo minimizar el gasto económico del mantenimiento de la infraestructura.

Esto es una gran ventaja, ya que es fundamental para este proyecto el hecho de que el despliegue sea rápido y escalable según la evolución del mismo, pero sin tener un coste inicial elevado, por lo que la implantación en esta infraestructura de tipo *cloud computing* es la mejor opción.

Además, el control de los clientes no ha sido especificado en el diseño, porque el objetivo es que esto lo realice directamente Google App Engine, ya que aporta varios servicios para el control de autenticación: "Firebase Authentication", "acceso con Google" u "OAuth 2.0 y OpenID Connect". Respecto a la elección entre las variantes que oferta, se ha escogido la última, porque permite la gestión y uso de tokens de autenticación desde cero con el mayor nivel de personalización.

5.1.3 Almacenamiento de datos

Todas las llamadas son de consulta, por lo que a priori no parece necesario hacer uso de ninguna base de datos, ya que a simple vista el cliente no guarda datos en la API propuesta, sino que este será quien deberá guardarse la información que se le proporciona para usarla y analizarla según su propósito. Pero como ya se ha podido ver en las funcionalidades, se guardan los datos de auditoría, por lo que sí se precisa de una base de datos y se prevé que se trabajará con grandes volúmenes de datos, ya que son todas las entradas y salidas de las consultas realizadas a la API, además de los errores que hayan ocurrido en el proceso.

Aunque el tipo más popular de base de datos sea las relacionales (SQL), en este proyecto se ha optado por el uso de las no relacionales (NoSQL) por todos los beneficios que puede aportar al mismo respecto a lo previsto, ya que este tipo está optimizado especialmente para aplicaciones que requieren grandes volúmenes de datos, baja latencia y modelos de datos flexibles. Además, uno de los tipos de almacenamiento de la información es como documento JSON, es decir, de igual manera en la que se recibe y se devuelve en las llamadas a esta API, según ha sido diseñada. Otra forma de almacenarla es como clave-valor que, al ser altamente divisibles, permiten escalado horizontal a escalas que otros tipos de bases de datos no logran alcanzar.

En general, sus beneficios se pueden resumir en cuatro puntos: [31]

- Desarrollo rápido e iterativo, gracias a los esquemas flexibles.
- Gran escalabilidad, ya que en lugar de escalar añadiendo servidores físicos y más caros, está diseñada para hacerlo empleando clústeres distribuidos de *hardware*.
- Mayor rendimiento, gracias a que están optimizadas para patrones de acceso y modelos de datos específicos.
- Proporcionan tipos de datos diseñados específicamente para cada uno de sus respectivos modelos y que las API sean altamente funcionales.

Por último, también es compatible con el uso de Google App Engine, sitio propuesto para alojar la API, ya que esta plataforma incluye el soporte nativo para Google Datastore [32], siendo esta una base de datos NoSQL muy escalable, como ya se ha comentado. Esta se encarga de manera automática de fragmentar y replicar, lo cual también favorece su durabilidad, y ofrece innumerables funciones similares a las consultas tipo SQL, haciéndolo más sencillo para quien no esté tan familiarizado con estas.

5.2 Estrategia de monetización

Esta API mayormente, aunque no únicamente, está enfocada para que la utilicen los analistas en ciberseguridad como clientes, pudiendo ser de manera individual o formando parte de una empresa, y las mismas redes sociales, aumentando así la seguridad que estas aportan. Por lo tanto, ahí se obtendría la mayor fuente de ingresos que se pretende tener para poder iniciar su desarrollo. Después, el mantenimiento se llevaría a cabo gracias a las suscripciones, ya que no será de uso libre completamente, sino que si se quiere usar para un fin profesional se deberá pagar en función del uso que se le va a dar. Además de la inversión extra que debería hacer una red social en el caso de que quisiera ser añadida a las distintas funcionalidades de la API.

De esta manera, se consigue que las personas de forma individual puedan probarla para ciertas consultas, que les ayuden a identificar fraudes en redes sociales para su uso personal, pero que las empresas o aquellas personas que trabajen tanto para las redes sociales como las que se benefician a partir de estas, sean quienes mantengan la API. Esto será controlado gracias a que se guardarán todas las consultas realizadas, pudiendo controlar qué usuario la ha realizado y restringirle el número de llamadas que pueda efectuar, obligando a que paguen en caso de que no les sea suficiente la cantidad que se oferte gratuitamente.

5.3 Planificación del proyecto

Para acabar con el apartado de aplicabilidad, propongo el uso de una metodología ágil de trabajo para desarrollar este proyecto, teniendo en cuenta que lo ideal sería que este proyecto no fuese implementado de manera individual, sino por un equipo de desarrolladores, con el fin de llevarlo a cabo en menos tiempo.

Esta elección se debe al hecho de estar familiarizada con esta manera de trabajar gracias a haberlas estudiado en el Grado en Ingeniería Informática y haber tenido la oportunidad de ponerlas en práctica en muchos trabajos grupales tanto del grado como del máster, en las prácticas de empresa realizadas en el grado e incluso en mi actual vida laboral. Además, esta experiencia fue satisfactoria, sobre todo con el uso de Scrum.

Esta manera de trabajar se utiliza para trabajos colaborativos, coincidiendo con el caso del proyecto de la API propuesta, consiguiendo así que los trabajos se llevase a cabo de una manera más ágil y rápida. Más específicamente, Scrum se caracteriza por realizar iteraciones, es decir, entregas parciales que se realizarán de forma regular, a corto plazo y con un resultado totalmente funcional, llamadas *sprints*, priorizando las funcionalidades que aporten más para el objetivo final del producto.

Esto se realiza para poder revisar, reflexionar, obtener *feedback* y propuestas de cambio sobre cada una de ellas, antes de que sea demasiado tarde para reconducir o reenfocar alguna parte sin que afecte en exceso al tiempo estimado en la planificación. [33]

Para organizar todas las tareas de forma explícita y visual se puede hacer uso de un tablero en el que se coloquen en distintas columnas, pudiendo usar Trello. Esta es una herramienta muy útil para ello, cuyo acceso es gratuito, además de haberla usado en asignaturas del Grado en Ingeniería Informática y en asociaciones a las que pertenezco, teniendo ya los conocimientos necesarios para trabajar cómodamente con ella. Un punto clave es que es muy cómoda para trabajar en equipo de una forma colaborativa, mejorando así la productividad y la organización del mismo. Cada tarea se representa en una tarjeta en la que se pueden incluir descripciones, listas, imágenes, archivos, enlaces, fecha de finalización, comentarios y una persona responsable. Después, a medida que se vaya avanzando en la misma se debe actualizar su estado, esto se consigue simplemente con arrastrar la tarjeta entre las columnas del tablero, explicadas a continuación.

Para comenzar se completa la primera columna, es decir, la pila de producto que se titula *Product Backlog*, ya que es donde se colocan todas las tareas que se pretenden llevar a cabo en el proyecto global. Cuando se comienza una iteración se escogen tareas de dicha columna y se pasan a la siguiente, *Sprint Backlog*, y se detalla todo el resto de la información que se puede aportar en la herramienta, siendo indispensable el reparto de tareas, asignando la o las personas responsables de cada una. Cuando una persona comienza con su tarea, debe pasarla a la columna titulada *In Progress*, donde permanecerá mientras se esté llevando a cabo el desarrollo de la misma. En cambio, cuando esté en periodo de pruebas, la tarea debe estar en la columna *In Test*. Por último, una vez se haya comprobado su correcto funcionamiento, ya se podrá colocar en la columna titulada *Done*. En esta última columna se irán acumulando las tareas realizadas, así que podrá usarse la opción de archivarlas, si se considera necesario, al menos en aquellas que sean de periodos anteriores.

Si estas indicaciones se siguen de manera correcta, todos los miembros del equipo pueden estar siempre al tanto del estado de las tareas y, por lo tanto, del proyecto. Además, como en la tarea se queda guardado todo el historial de cambios realizados en ella, se podrá revisar el flujo que ha seguido. Aunque esa no es la única utilidad, sino que hay más, como por ejemplo saber a quién preguntar sobre alguna tarea, viendo la o las personas que la tengan asignadas, en el caso de que fuese necesario.

Adicionalmente, aunque prácticamente todo esté en el tablero, la comunicación es indispensable, así que esta metodología indica que se deben realizar una serie de reuniones de manera periódica.

Diariamente existe la llamada *daily*, es decir, una reunión de corta duración en la que cada integrante del equipo debe decir de manera escueta lo que hizo y qué inconvenientes se encontró el día anterior, y lo que pretende realizar ese mismo día y qué dependencias o qué necesita para ello. El resto de reuniones son con menos frecuencia, ya que son al inicio (*sprint planning*) y al final (*sprint review*) de cada iteración, para tratar respectivamente la planificación de la misma y realizar una retrospectiva, con el fin de sacar conclusiones que ayuden a mejorar en los siguientes.

Por último, en esta metodología existen distintos roles asignados a los profesionales del equipo de trabajo: desarrollador, scrum master (gestiona el proceso y evita que los desarrolladores tengan impedimentos) y project manager (maximiza y optimiza el valor del producto, además de ejercer de portavoz del equipo ante los clientes, con todo lo que ello conlleva). El rol asignado a más integrantes del equipo es el de desarrollador, aunque todos ellos son igual de importantes e imprescindibles.

6. Conclusiones

Este proyecto me ha permitido analizar, estudiar e indagar más de lo que lo había hecho hasta el momento sobre los tipos y ejemplos reales de *malware*, las funcionalidades accesibles de las redes sociales y la relación entre ambas.

Todo ello ha sido útil para el diseño que se ha realizado, en este caso, de una API que está pensada para dar la información que ayude a identificar *botnets* maliciosas u otros fraudes en las redes sociales, cuyas funcionalidades han sido estudiadas y especificadas. Esta, por el momento, aporta información obtenida desde la que se puede conseguir de manera pública sobre Twitter e Instagram, pero más concretamente aquella que se ha visto útil para el propósito ya comentado.

Pero el punto fuerte de este proyecto no es la información que aporta en sí, sino para lo que esta puede ser usada a posteriori, ya que se podría actuar sobre aquello que se identifique y mejorar la seguridad de la red social en la que se encuentre. Además, su desarrollo se llevará a cabo mediante el uso de librerías de Python que lo facilitan, al menos para las redes sociales indicadas, siempre sin afectar ni la privacidad ni seguridad de sus usuarios.

6.1 Dificultades

Quisiera destacar todo el esfuerzo invertido en la investigación realizada en este trabajo de fin de máster ya que, aunque a priori puede parecer que es posible encontrar mucha información al respecto, la mayoría se repite y no siempre es totalmente correcta ni clarificadora respecto a detalles más técnicos, que era lo que se buscaba.

También, aunque en menor medida, ha sido un reto el hecho de haber trabajado en Python, siendo este un lenguaje de programación que, como ya he comentado en la explicación de su elección, no he usado en ninguna ocasión anterior a este trabajo. Aunque por el momento no se ha realizado el desarrollo, sino solo las pruebas a la API de Instagram con la librería Instabot, así que esta dificultad será mayor cuando se desarrolle el proyecto, pero será tomado como un reto y aprendizaje.

6.2 Líneas futuras

Este apartado se incluye con el fin de poder comentar y, por lo tanto, tener en cuenta todo aquello que no se ha podido abarcar en este trabajo.

La línea futura principal sería la integración con *machine learning*, habiendo sido pensado y diseñado ya con dicho objetivo; pero, para eso, previamente se debe realizar el desarrollo de todo lo expuesto en este trabajo y después su puesta en marcha, por lo que se tomará como prioridad.

Posteriormente, la intención es que esta API abarque el mayor número de redes sociales que sea posible, así que a pesar de comenzar con Twitter e Instagram, se procederá al aumento de estas, incluyéndolas a cada una de las funcionalidades. Aunque conllevará el estudio de cada una de las redes sociales que se vayan a incluir, esto será posible gracias al gran potencial de este proyecto por haberse diseñado de manera flexible en prácticamente todos los aspectos.

También sería interesante realizar pruebas de rendimiento una vez esté realizado el diseño inicial, ya que este proyecto necesitará un amplio ancho de banda y hay muchos detalles al respecto que no se pueden saber con exactitud anteriormente y, por lo tanto, no se puede prever un diseño totalmente óptimo. Con estas pruebas, se podrían realizar adaptaciones para mejorar los tiempos y, sobre todo, para evitar que no se termine el tiempo de espera de la solicitud.

Por último, se pretende que las funcionalidades de la API aumenten, a medida que sea posible y se tenga la necesidad, pudiendo también atender peticiones de los propios clientes y de las redes sociales, evolucionando en función del progreso de las mismas. Aunque, para ello, no se deben solo agregar funcionalidades nuevas, sino que es imprescindible ir adaptando las ya existentes, para que no queden desfasadas respecto a la red social.

7. Referencias

[1] Vilatec, «Vilatec Information Technologies». *https://www.vilatec.com/tipos-de-malware-y-anti-malware/*

[2] Mike, «YouTube», F-Segure en Finlandia, 10 Marzo 2011. *https://www.youtube.com/watch?v=lnedOWfPKT0&t=317s*

[3] «Wikipedia», 24 Junio 2021. *https://es.wikipedia.org/wiki/(c)Brain*

[4] «Portaltic», 18 Junio 2021. *https://www.europapress.es/portaltic/ciberseguridad/noticia-malware-antipirateria-impide-acceder-paginas-descarga-ilegal-contenidos-software-20210618150654.html*

[5] T. Meskauskas, «PCrisk», 13 Enero 2021. *https://www.pcrisk.es/guias-de-desinfeccion/8833-bitcoinminer-trojan*

[6] «EcuRed». *https://www.ecured.cu/Virus_inform%C3%A1tico_I_Love_you#Caracter.C3.ADsticas*

[7] Y. González, «Grupo Atico34», 17 Septiembre 2020. *https://protecciondatos-lopd.com/empresas/gusano-informatico/*

[8] F. Ramirez, «Derecho de la red», 18 Diciembre 2019. *https://derechodelared.com/slammer-el-virus-que-colapso-internet-en-2003/*

[9] Redacción, «RÍO NEGRO», 10 Marzo 2011. *https://www.rionegro.com.ar/el-primer-gusano-de-la-ciberguerra-ETRN_577704/*

[10] J. Fruhlinger, «CSO», 22 Agosto 2017. *https://www.csoonline.com/article/3218104/what-is-stuxnet-who-created-it-and-how-does-it-work.html*

[11] B. Donohue, «Kaspersky», 21 Octubre 2013. *https://www.kaspersky.es/blog/cuatro-ejemplos-de-troyanos-bancarios/1696/*

[12] Redacción, «Computing», 22 Julio 2021. *https://www.computing.es/seguridad/noticias/1127382002501/espana-principal-pais-europeo-objetivo-de-familia-de-malware-bancario-tetrade.1.html*

[13] «EuropaPress Portaltic», 5 Julio 2021. *https://www.europapress.es/portaltic/ciberseguridad/noticia-google-play-elimina-apps-android-robaban-contrasenas-facebook-20210705125644.html*

[14] K. Team, «Kaspersky», 26 Abril 2021. *https://www.kaspersky.es/blog/top5-ransomware-groups/25126/*

[15] J. Jiménez, «Redes Zone», 17 Diciembre 2020. *https://www.redeszone.net/tutoriales/seguridad/scareware-amenaza-seguridad/*

[16] «Software Lab». *https://softwarelab.org/es/que-es-spyware/*

[17] «Kaspersky». *https://www.kaspersky.es/resource-center/threats/riskware*

[18] «Ayudaley». *https://ayudaleyprotecciondatos.es/2021/04/28/riskware/#Ejemplos_de_software_de_riesgo*

[19] L. Zeltser, «Lenny Zeltser»,14 Febrero 2015. *https://zeltser.com/bots-control-social-networking-content/*

[20] P. F. Iglesias, «PabloYglesias», 2020. *https://www.pabloyglesias.com/uso-botnets-viralizar-contenido/*

[21] S. Ghosh, G. Korlam y N. Ganguly, «Spammers' networks within online social networks: a case-study on twitter», Department of CSE, Indian Institute of Technology Kharagpur, India, Hyderabad, India, 2011.

[22] C. Yang, R. Harkreader, J. Zhang, S. Shin y G. Gu, «Analyxing spammers' social networks for fun and profit - a case study of cyber criminal ecosystem on twitter», Lyon, France, 2012.

[23] S. Ghosh, B. Viswanath, F. Kooti, N. K. Sharma, G. Korlam, F. Benevenuto, N. Ganguly y K. P. Gummadi, «Understanding and Combating Link Farming in the Twitter Social Network», Lyon, France, 2012.

[24] J. Zhang, R. Zhang, Y. Zhang y G. Yan, «The Rise of Social Botnets: Attacks and Countermeasures», 2016.

[25] «Twitter Developer Platform». *https://developer.twitter.com/en/docs*

[26] «Twitter», 2021. *https://help.twitter.com/es/rules-and-policies/twitter-api*

[27] U. Harmon758, «Github», 26 Diciembre 2020. *https://github.com/tweepy/tweepy/tree/29df541c1f72dee6b2ee7a9d71e21e90f7634b13*

[28] «FACEBOOK for Developers». *https://developers.facebook.com/docs/instagram-api*

[29] «WeLiveSecurity», 9 Octubre 2014. *https://www.welivesecurity.com/la-es/2014/10/09/exploits-que-son-como-funcionan/*

[30] «Programmer Clieck». *https://www.programmerclick.com/article/7009969406/*

[31] «Amazon». *https://aws.amazon.com/es/nosql/*

[32] *«Google Cloud». https://cloud.google.com/datastore?hl=es*

[33] K. S. Rubin, Essential Scrum: A Practical Guide to the Most Popular Agile Process, Addison-Wesley, 2012.